**MASTER EN SCIENCES INFORMATIQUES**

# Utilité d'un dictionnaire de données pour l'union des classes moyennes à Namur Conception et implémentation

Dols, Philippe; Leger, Christophe

*Award date:*
1985

*Awarding institution:*
Universite de Namur

**FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX**
**NAMUR**

INSTITUT D'INFORMATIQUE

# UTILITE D'UN DICTIONNAIRE DE DONNEES POUR L'UNION DES CLASSES MOYENNES A NAMUR.

## Conception et implémentation.

Mémoire présenté par

**Philippe DOLS** et
**Christophe LEGER**

en vue de l'obtention du grade de
Licencié et Maître en Informatique

---

Année académique 1984-1985

## AVANT-PROPOS.

Nous remercions chaleureusement toutes les personnes qui, de près ou de loin nous ont permis de réaliser ce projet.

Nous sommes particulièrement redevables à Monsieur le Professeur Hainaut qui a proposé le titre de ce travail. Pour l'intérêt qu'il a porté à son élaboration, pour les conseils qu'il nous a donnés et qui ont guidé notre recherche ainsi que pour sa constante disponibilité, nous lui exprimons notre profonde gratitude.

Monsieur Saint Amant, Directeur du Service informatique de l'UCM, nous a initiés aux principes de fonctionnement de son entreprise. De son souci de mettre à notre disposition toutes les ressources nécessaires, de son obligeante sollicitude, nous lui sommes profondément reconnaissants.

A Colette Simenon, Françoise Leger et Anita Voz, nous adressons tous nos remerciements pour le travail ingrat de dactylographie, de dessin de schémas, d'assemblage et d'impression.

TABLE DES MATIERES
---

CHAPITRE 5 : ETUDE DE L'IMPLEMENTATION DU DD



CHAPITRE 6 : MANIPULATION DU DD

## ORGANISATION DU TRAVAIL.

Ce travail a été conçu et réalisé à deux. Il est le fruit d'une combinaison d'idées, de recherches, de propositions ainsi que d'un dialogue permanent et critique. Il nous a permis d'appréhender la complexité de la réalité informatique et de mieux comprendre les principes de base qui nous ont été enseignés pendant ces deux années d'études à l'Institut d'Informatique des Facultés Notre-Dame de la Paix de Namur.

## LECTURE RAPIDE.

Nous suggérons au lecteur pressé la lecture des chapitres 3 et 4 qui permet de comprendre l'essentiel de notre démarche.

## ABREVIATIONS.

| | |
|---|---|
| BD: | Base de Données. |
| CAS: | Caisse d'Assurances Sociales. |
| DD: | Dictionnaire de Données. |
| E.A.: | Entité-Association. |
| FNDP: | Facultés Notre-Dame de la Paix. |
| MAG: | Modèle d'Accès Généralisé. |
| MRG: | Modèle Relationnel Généralisé. |
| O.S.: | "Operating System". |
| SDD: | Système de Dictionnaire de Données. |
| SGBD: | Système de Gestion de Base de Données. |
| SGD: | Système de Gestion de Données. |
| SQL: | "Search Query Language". |
| SQL/DS: | "Search Query Language/Data System". |
| T.A.: | Type d'Association. |
| TAS: | Type d'Association. |
| T.E.: | Type d'Entité |
| UCM: | Union des Classes Moyennes. |

# INTRODUCTION.

L'objectif de ce travail consiste à concevoir et implémenter un Dictionnaire de Données (DD) en vue d'en démontrer son utilité dans le cadre de l'Union des Classes Moyennes (UCM) de Namur.

Pour ce faire, nous expliquerons l'utilité générale des Systèmes de Dictionnaire de Données (SDD) et établirons leurs caractéristiques principales (complétude, cohérence, convivialité, extensibilité, intégration, portabilité).

Nous introduirons ensuite le lecteur dans l'organisation de l'UCM en caractérisant plus spécialement les outils de documentation disponibles (le dossier informatique, les tables système SQL).

Cette étape nous permettra de mettre en évidence les besoins supplémentaires en documentation à l'UCM. Celle-ci devra notamment couvrir toutes les étapes du cycle de vie du SI propre à l'UCM et permettra de décrire des relations de correspondance (ou "mapping") entre différentes descriptions particulières.

Nous essayerons de montrer la puissance potentielle de l'outil SDD et plus particulièrement l'adéquation de cet outil aux besoins documentaires de l'UCM en vue d'une meilleure gestion de sa ressource information.

Dans ce but, nous implémenterons sur le site même de l'UCM (IBM 4361) un prototype SDD dont nous fixons à priori certaines caractéristiques limitatives (SDD passif, dépendance SGBD de type application, extensibilité possible, contrôle de cohérence prévu mais non implémenté, complétude possible, convivialité minimale).

Nous concevrons un modèle Entité Association (E.A.) décrivant le plus généralement possible tout système d'informations et en particulier celui de l'UCM.

Nous l'implémenterons en deux temps.

Dans un premier temps, nous concevrons une structure d'accueil intermédiaire entre le modèle et le SGBD afin de satisfaire l'exigence d'extensibilité formulée au départ.

Nous procéderons ensuite à l'implémentation proprement dite au moyen du SGBD SQL en veillant autant que possible à maintenir cachée de l'utilisateur la structure d'accueil intermédiaire.

Nous implémenterons ensuite quelques outils de chargement initial et de maintenance dans le cadre restreint de la dépendance forte SGBD SQL. Un programme de contrôle de cohérence partiel lors des manipulations sur le DD sera spécifié.

Reste à évaluer dans quelle mesure le dictionnaire implémenté répond aux objectifs d'outil de gestion de la ressource information dans le cadre de l'UCM. Dans cette optique, nous évaluerons la capacité documentaire du SDD ainsi que sa facilité d'exploitation.

A titre démonstratif, nous introduirons la descriptions de deux applications test(le SDD lui-meme et le sous-module ASE du service CAS de l'UCM).

# CHAPITRE 1.

# UTILITE DES SDD.

## 1.1. INTRODUCTION.

La reconnaissance de caractère vital pour toute organisation de sa ressource en informations conduit à la nécessité de la contrôler au même titre que toute autre ressource financière, matérielle ou humaine.

En effet, la qualité du fonctionnement tant interne qu' externe d'une organisation, la capacité des gestionnaires à évaluer les performances, à prévoir, planifier et réagir aux circonstances dépend pour une large part de la disponibilité rapide d'informations pertinentes, claires et exactes.

D'autre part, la disponibilité d'une documentation standardisée et automatisée couvrant tous les aspects du traitement de l'information est apparue comme une nécessité devant permettre le contrôle et la gestion de la ressource en information. Celle-ci est notamment destinée à corriger les insuffisances de la documentation manuelle que Teichroew, Rataj, Hershey (8) résument de la sorte :

- temps perdu considérable quand on change quelque chose dans la description du problème,
- mode de travail essentiellement séquentiel, négligeant les possibilités de parallélisme des activités,
- risque important d'erreurs difficilement décelables,
- problèmes de coordination entre les personnes collaborant au projet,
- validation quasi impossible de la solution retenue et décrite.

Enfin, l'informatisation de plus en plus large des organisations a imposé la mise en place progressive d'une gestion de la ressource "information" tant en raison de l'évolution des techniques (traitements en temps réel) que de l'investissement consenti.

Nous allons illustrer l'évolution de ce processus en terme de structuration du sous-système informatisé du système d'information (SI) définie à partir des différents services l'utilisant.

La figure 1 décrit le modèle général d'un SI tel qu'il est défini dans (11)

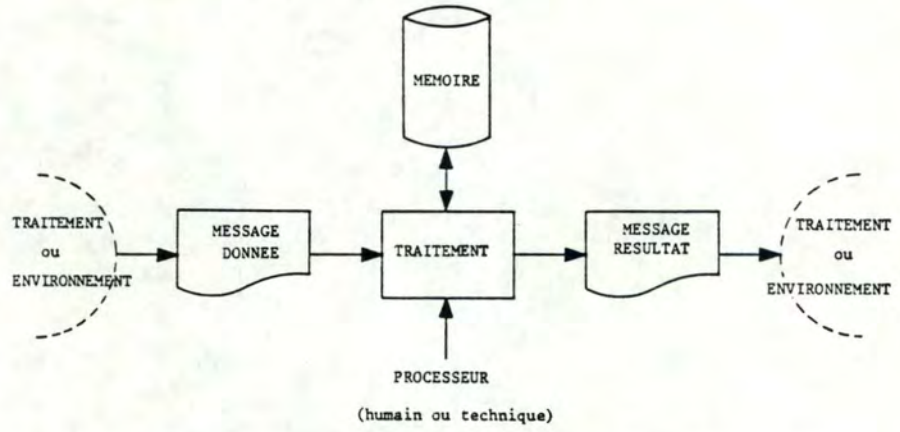



*Figure 1 : Schéma du modèle général de tout système d'information.*

"...Un message-donnée, en provenance d'un traitement ou de l'environnement du SI est communiqué au SI qui le traite via un processeur, éventuellement à l'aide de sa mémoire, pour engendrer un message-résultat, lequel est à son tour transmis à un autre traitement ou à l'environnement du SI..."(BODART)

### 1.1.1. Structure de première génération (figure 2)

Les services s'automatisent individuellement et possèdent chacun leurs fichiers et leurs programmes propres.

Cette structure définit une partition selon les différents services qui utilisent le SI. Elle implique un maximum de redondances tant au niveau des données que des traitements.

La documentation correspondante est typiquement :
- manuelle
- non standardisée
- peu contrôlée
- propre à chaque service.

### 1.1.2. Structure de deuxième génération.(figure 3)

L'avènement des traitements en temps réels a amené les entreprises à acquérir des SGBD. Dans un premier temps, ceux-ci ont été mis en oeuvre comme de simples méthodes d'accès sophistiqués. Cette structure n'entraine pas de changements fondamentaux par rapport à l'organisation précédente.

### 1.1.3. Structure de troisième génération (figure 4).

Dans un deuxième temps, les SGBD ont été mis en oeuvre comme de véritables outils de partage des données (approche base de données). La transition vers cette nouvelle structure a nécessité la mise en place d'une gestion de la ressource "données" et la redéfinition du S.I. Elle a vu apparaitre une nouvelle fonction : celle d'administration des données.

Cette structure implique un contrôle des redondances, un début de standardisation de la documentation qui reste cependant largement manuelle.

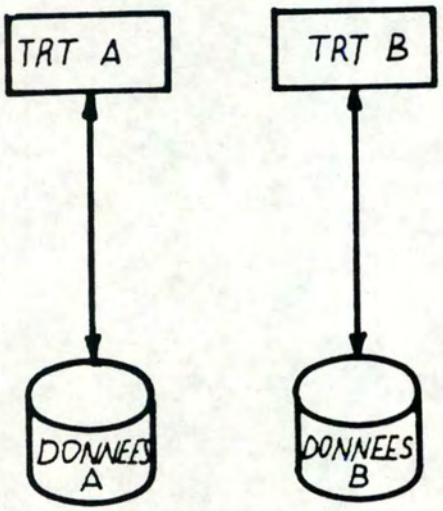



figure 2

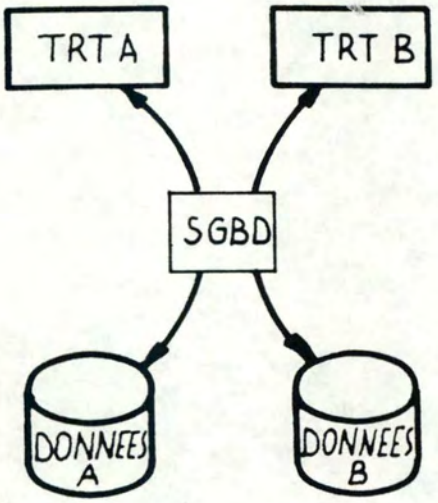



figure 3

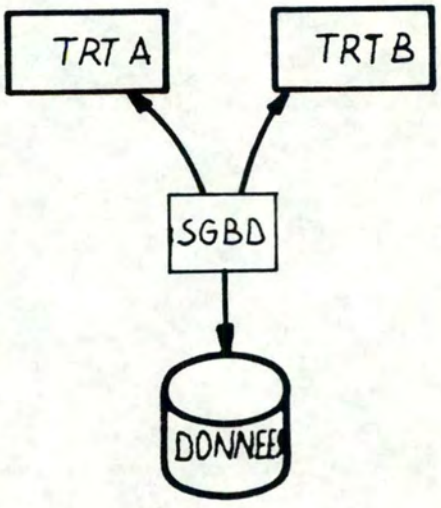



figure 4

### 1.1.4.Structure de quatrième génération (figure 5)

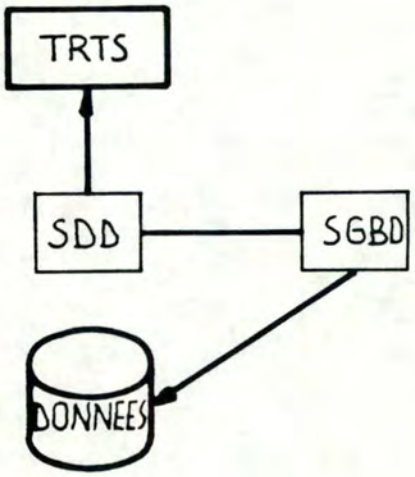



figure 5

Certaines méthodologies de développement de logiciels produisent des architectures composées de modules structurés en réseau. Il est dès lors possible de définir les traitements indépendamment les uns des autres.

Ainsi, les régles de validation de données (PICTURE de COBOL) seraient stockées une fois seulement et activées lors des opérations d'insertion ou de mise à jour des données.

Il en découle :
- contrôle des redondances au niveau traitement
- contrôle des traitements et renforcement de standards
- travail de programmation simplifié
- accroissement du traitement parallèle.

Cette structure implique la mise en place d'un outil documentaire centralisé et automatisé, combinant les descriptions de données et traitements utilisés par l'organisation et qu'il est convenu d'appeler SYSTEME DE DICTIONNAIRE DE DONNEES (SDD).

## 1.2. Définition d'un SDD.

### 1.2.1. Objectifs d'un SDD

#### Améliorer la documentation

Le SDD est appelé à servir de référence documentaire pour tout utilisateur du SI. Il permet une production de documentation plus facile, standardisée, propre à l'utilisateur.

Bénéficiaires : tous les utilisateurs du SI

#### Contrôler les données

Le SDD centralise les informations spécifiant l'usage des données. En contrôlant le DD, il est possible de contrôler l'usage effectif des données.

Bénéficiaires : administration des données, analyste.

#### Contrôler la conception et le développement des applications

Par la documentation qu'il enregistre et fournit, spécifiant le projet et ses différentes étapes en élaboration, le SDD permet de mieux gérer le développement des applications.

Bénéficiaires : administration des donnée, programmeur.

#### Mesurer l'impact d'une modification

Le SDD est capable d'évaluer l'impact d'un changement en temps et en coût.

Bénéficiaires : gestionnaires de l'entreprise, administration de données, analystes.

### 1.2.2. Composants d'un SDD

Un SDD est une application comprenant :

une base de données (le dictionnaire de données: DD)

Le DD traduit le modèle du SI. Les données (ou occurrences) qu'il contient décrivent le fonctionnement de l'organisation. Elles sont donc d'un niveau d'abstraction supérieur à celui des données traitées par le SI. On parlera de méta-données.

des fonctions

telles que :

- maintenance
  ces fonctions interprètent et traitent les requêtes d'insertion, de modification, de suppression du contenu du DD.

- traitement d'interrogation
  Ces fonctions permettent d'interroger le DD dans un langage proche du langage naturel et d'en extraire des méta-données.

- génération de rapports
  Ces fonctions fournissent des rapports prédéfinis plus ou moins sophistiqués (résumé d'état d'avancement d'un projet, rapports sous forme matricielle, rapports graphiques...).

- gestion
  Ces fonctions assurent les contrôles de sécurité, intégrité, concurrence, reprise sur incidents...

- interface "software"
  Ces fonctions fournissent des méta-données formattées (des résultats "software") en réponse à des commandes de génération (requêtes "software")
(cfr figure 6 ).

- conversion
  Ces fonctions parcourent les programmes sources et produisent automatiquement des transactions de maintenance facilitant de la sorte le chargement initial du DD.

Il reste à l'administration des donnnées le soin de contrôler les transactions générées afin de repérer redondances ou entrées non standard.

## 1.3. CARACTERISTIQUES D'UN SDD

### 1.3.1 Complétude

Le DD doit contenir une description complète du SI couvrant toutes les étapes de son cycle de vie, discriminant les sous-systèmes en exploitation de ceux en développement.

Il s'élabore progressivement au fur et à mesure de la conception du système, depuis le niveau conceptuel jusqu'au niveau opérationnel.

De la richesse des informations qu'il contient dépendra l'étendue de son champ d'action ainsi que le type d'utilisation qui en sera fait. Le SDD intégrera différents niveaux de description du SI pouvant concerner des types d'utilisateurs spécifiques.

D'autre part, il importe que le SDD permette d'enregistrer les différentes communications entre utilisateurs. Le contrôle et la documentation des communications peut s'opérer en définissant des correspondances (ou mapping) entre différents niveaux de description ou entre memes niveaux de description.

### 1.3.2. Cohérence - validité

Pour qu'un SDD remplisse son rôle premier d'outil documentaire, il doit non seulement être complet mais encore correct.

De la validité des informations qu'il contient dépendra la confiance qu'il inspirera et donc l'utilisation effective qui en sera faite.

Les fonctions de controle de cohérence assurent la cohérence du modèle du SI lors des opérations d'insertion ou de suppression des méta-données.

Ce contrôle peut s'opérer à trois niveaux :
- modèle général du SI
- sous-modèle correspondant à un niveau de description donné
- mapping entre sous-modèles.

### 1.3.3. Convivialité

De la facilité d'accès et de manipulation du SDD dépendra l'usage qui en sera fait.

Un SDD "convivial" assurera des fonctions classiques de manuel d'emploi et de guidage de l'utilisateur.

### 1.3.4. Indépendance vis-à-vis d'une structure d'organisation.

Les organisations diffèrentes en structure et en besoin et l'usage qu'il sera fait d'un SDD variera avec la structure d'une organisation.

Un SDD doit contenir le modèle de base de toute organisation; il doit pouvoir être étendu en vue de représenter une organisation spécifique.

### 1.3.5 Intégration à l'environnement automatisé

#### SDD de première génération.

Le SDD passif constitue un simple outil de documentation automatisé, destiné à pallier les inconvénients de la documentation manuelle et sans interaction avec son environnement automatisé. Par l'image qu'il donne du SI, il fournit une aide précieuse à la gestion de la ressource information.

Cependant,il est peu probable qu'un SDD isolé soit capable de contenir tous les détails de toutes les données utilisées dans l'organisation (exigence de complétude). Il est donc important que les méta-données contenues dans les programmes sources, SGBD, tables superviseurs de télétraitements, etc... soient converties dans le SDD dans le format requis (fonction de conversion).

De plus, il est souhaitable que les méta-données du SDD puissent être livrées dans le format requis au programme ou système qui les utilisent, si l'on veut

imposer via le SDD des standards de descriptions de données à toute l'organisation (fonction interface "software").

Enfin, le SDD a plus de chances d'être à jour si toutes les manipulations et définitions de méta-données automatisées transitent par lui.

SDD de deuxième génération.
---

Le SDD actif interagit avec son environnement automatisé au moment de la compilation (préprocesseur).

Ainsi, tout accès ou mise à jour de données dans la BD ou dans un fichier sera contrôlé à la compilation à l'aide des méta-données contenues dans le SDD.

De la sorte, le SDD garantit la validité et la cohérence des données manipulées par le systéme.

Une utilisation courante du SDD par des programmes concerne l'inclusion de méta-données à la compilation. La figure 6 reprend un exemple de génération dans le format COBOL des méta-données requises à partir de la commande GENERATE tel qu'il est décrit dans (5). Des capacités d'édition ou des options de format peuvent rendre l'interface plus souple.

De plus, le SDD peut jouer un rôle actif dans la conception et le développement des applications à partir d'un ensemble intégré d'outils interfaçant son dictionnaire.

Il est en effet possible d'automatiser partiellement les traitements liés à certaines opérations de "mapping" , l'optimisation des schémas objets obtenus demeurant une tâche manuelle.

Les multiples outils existant couvrent les différentes étapes du cycle de vie d'un projet (spécification, conception, maintenance).

Le lecteur intéressé en trouvera une description panoramique dans ( 7 ).

Le SDD actif joue un rôle central de référence et de contrôle dans toute l'organisation automatisée.

Il facilite et rend plus sûre la conception et le développement des applications en en automatisant certaines étapes ou en fournissant des outils de documentation plus puissants.

(a)
```
GENERATE PAYMENT-FILE
    INSERT "PAY" BEFORE "CUST"
    REPLACE "NUMBER" WITH "NUM"
    INCLUDE COMMENTS
    INCREMENT LEVEL-NUMBERS BY 2
    INDENT 3 SPACES.
```

(b)
```
FD IN-PAYMENT-FILE
 RECORD CONTAINS 50 CHARACTERS
 BLOCK CONTAINS 20 RECORDS
 VALUE OF ID "PFILE ØØ1"
***This is a transaction file for the
***Customer Credit System
 Ø1 PAYMENT-RECORD.
   Ø3 PAY-CUST-NUM              PIC 9(10) COMP-2.
   Ø3 PAY-CUST-NAME.
      Ø5 PAY-CUST-LNAME         PIC X(20).
      Ø5 PAY-CUST-FNAME         PIC X(10).
      Ø5 PAY-CUST-INIT          PIC X.
   Ø3 PAY-CUST-DATE             PIC X(6).
   Ø3 PAY-CUST-AMOUNT           PIC 9(6)V99.
```

figure 6 Generate function example: (a) GENERATE command for COBOL; (b) generated COBOL file description.

SDD de troisième génération.
---

Le SDD dynamique interagit avec son environnement automatisé au moment de l'exécution.

Ici, l'effet du SDD est immédiat sur les transactions traitées alors que l'effet du SDD actif dépend de la recompilation éventuelle des programmes concernés.

1.3.6.Indépendance vis-à-vis d'un environnement automatisé
---

Idéalement, un SDD devrait pouvoir opérer dans tout environnement automatisé (configuration "hard" ou SGBD,...) sans altération de performances ou de fonctionnement.

Cet idéal est cependant loin d'etre réalisé.
La figure 7 illustre les trois approches de la dépendance entre un SDD et un SGBD, telles qu'elles sont décrites dans (5)

la dépendance faible:
---

Le SDD n'utilise aucun des utilitaires du SGBD; il prend en charge les fonctions de gestion du DD.
Avantage: le SDD peut procurer les fonctions de description, conversion et interface dans une multitude d'environnements caractérisés par différents SGBD ou pas de SGBD.

la dépendance forte:
---

Le DD constitue pour le SGBD une BD comme une autre, dont il prend en charge la plupart des fonctions de gestion.
Avantage: une seule fonction de gestion de BD (fournie par le SGBD).

la dépendance totale ("embedded")
---

Ici, le SDD est un composant du SGBD; il est totalement intégré au SGBD.
Avantage: une seule source de méta-donnée (le DD).

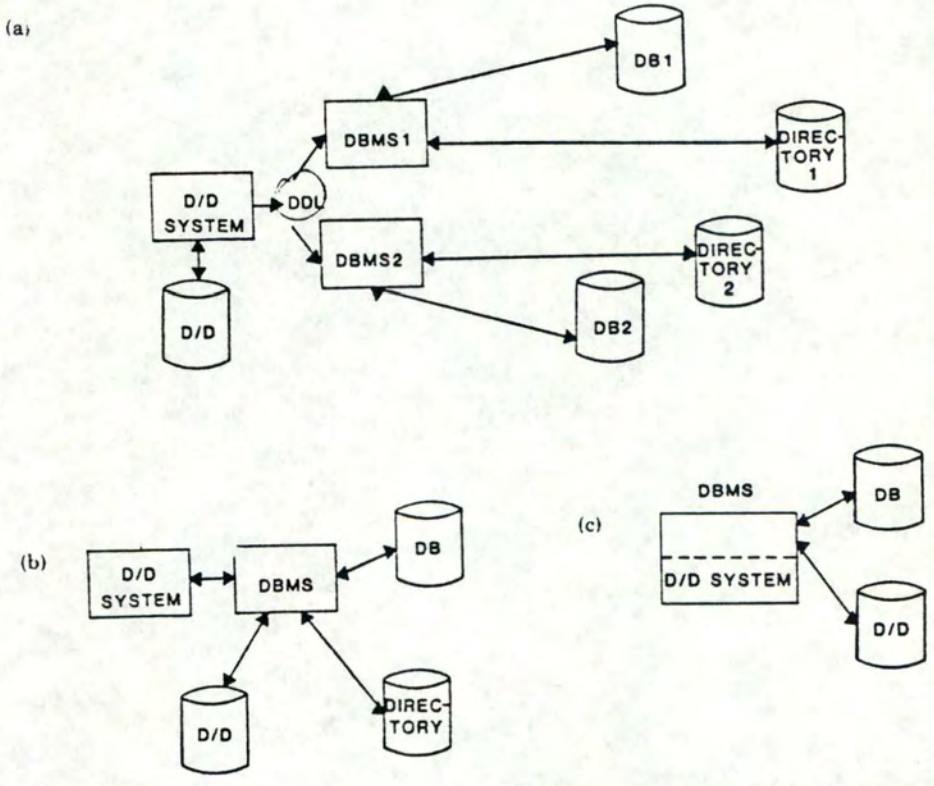



figure 7 D/D System: DBMS relationships. (a) Independent approach; (b) DBMS—application approach; (c) embedded approach.

# CHAPITRE 2

# CADRE GENERAL DE L'UCM.

## 2.1. ORGANISATION DES SERVICES A L'UCM.

L'Union des Classes Moyennes à Namur comporte 4 services distincts.

### 2.1.1.. Le Secrétariat Social (SS)

Le SS a pour mission:
- le calcul des salaires
- l'assistance juridique
- l'information sur la gestion des entreprises
- le secrétariat administratif
- la comptabilité

Environ 4.000 entreprises y sont affiliées (soit quelque 20.000 travailleurs.

### 2.1.2. La Caisse d'Assurances Sociales (CAS)

Elle a pour mission:
- la perception des cotisations sociales des travailleurs indépendants
- les prestations familiales
- les prestations d'assurance maladie-invalidité
- les prestations de retraite et de survie
- le secrétariat administratif
- la comptabilité

Environ 90.000 travailleurs indépendants des provinces de MONS et LIEGE y sont affiliés.

### 2.1.3. Caisse de compensation pour allocations familiales (CCAF)

La CCAF a pour mission principale la tenue des prestations familiales concernant les salariés des entreprises affiliées.

Environ 5.000 entreprises sont affiliées à la CCAF (soit quelque 20.000 virements par mois).

### 2.1.4. Programmes et Services (PME).

Les PME ont pour mission:
- le développement de programmes à la carte
- la tenue des comptabilités

Environ 50 entreprises sont affiliées.

## 2.2. CARACTERISTIQUE DU TRAITEMENT INFORMATIQUE

A chaque service correspond un centre de traitement informatique (CTI) prenant en charge la gestion des informations du service.

### 2.2.1. L'équipement informatique

Deux ordinateurs prennent en charge l'ensemble des traitements informatiques:

- la machine UNIVAC (progressivement abandonné), sur laquelle sont notamment implémentés les traitements suivants:
  - l'application CCAF (en conversion vers IBM)
  - SS. COMPTA (en conversion vers IBM)
  - l'application PME

- la machine IBM 4361 sur laquelle sont notamment implémentés les traitements suivants:
  - l'application CAS
  - SS. SALAIRES
  - SS. INFOGESTION
  - SS. INFOJURIDIQUE

Ils sont connectés à une cinquantaine de terminaux et à une dizaine de micros-ordinateurs.

### 2.2.2.La notion de machine virtuelle (MV)

L'OS VM (IBM 4361) attribue aux différents utilisateurs des ressources auxquelles correspond la notion de machine virtuelle (MV).
Toute personne d'un service peut travailler dans la MV d'un autre service.
Tout fichier physique et programme est défini sur une et une seule MV.

### 2.2.3.Les fichiers

Les fichiers sont implémentés sous plusieurs formes. On distingue les fichiers de type:

- DBS UNIVAC (B-D CODASYL)
- KSDS (fichiers indexés)
- CMS (fichiers séquentiels)
- DB (tables SQL)

D'autre part, un même fichier peut exister en plusieurs exemplaires pour raisons

- de sécurité
Ainsi, il est procédé chaque semaine au sauvetage par blocs physiques de tous les fichiers (sécurité vis-à-vis du crash disque). De même, il est procédé chaque jour à la fusion du fichier de base et du fichier mouvements du jour (un fichier SIGNALETIQUE mouvementé représente environ 2% du fichier de base)

- de traitement
Certains travaux spécifiques sont définis périodiquement sur certains fichiers. Il en est ainsi du fichier SALAIRE du jour par exemple qui sera recopié dans le fichier HISTORIQUE des paies du trimestre en vue d'opérations, de statistiques....

- d'archivage
IL est procédé en fin d'année à l'archivage sur bande magnétique de certains fichiers (fichier SALAIRE annuel...)

### 2.2.4. Les traitements

Les traitements opérés à l'UCM sont de type "batch" ou interactif
Ils sont structurés en:

- service (ou application): correspond à l'ensemble des traitements d'un service
- module:correspond à la notion d'application du modèle de structuration des traitements.
- sous-module (facultatif)
- opération: correspond à l'unité d'exécution
- programme: correspond à l'unité de compilation

## 2.3. ORGANISATION DE LA DOCUMENTATION DES APPLICATIONS.

### 2.3.1. Le dossier informatique

La documentation des traitements informatiques est essentiellement manuelle. En vue de sa standardisation, les CTI ont défini un DOSSIER INFORMATIQUE type qui est déjà opérationnel dans le service CAS. Le lecteur en trouvera des extraits significatifs en annexe 3.1.

Le dossier informatique contient, outre les recommandations d'usage:

- la description de la mission du service

- la présentation de l'application CAS

- la description de l'architecture physique de l'application (décomposition en modules) et la spécification concrète des modules.

- une brève spécification des sous-modules

- une brève spécification des opérations comprenant la description des écrans de dialogue liés aux programmes

- l'algorithme des opérations

- le code source des différents programmes liés à l'opération.

L'inventaire des différents fichiers ainsi qu'un index de composition des différents traitements sont par ailleurs fournis.

Le dossier est mémorisé sur support magnétique et en principe accessible à tout utilisateur du SI.

En pratique, il s'adresse principalement:

- au programmeur qui retrouve une documentation concernant les programmes physiques

- à l'analyste qui dispose d'une certaine base documentaire pour concevoir et développer les applications

- au responsable CTI qui supervise cette documentation et veille à sa tenue à jour en vue d'améliorer la gestion de la ressource information

- à l'utilisateur final qui peut,éventuellement recourir au dossier pour comprendre les différentes séquences d'écrans ainsi que les modalités du dialogue interactif.

Cependant, on ne trouvera aucune information dans le dossier concernant les différents stades de conception et de développement des applications.
Aucune correspondance n'étant dès lors possible entre les différents niveaux de description d'une application, Il s'en suit:
- une compréhension difficile de l'application
- peu d'amélioration au niveau de la gestion de l'application.

Par ailleurs, le dossier informatique comporte les inconvénients (signalés au paragraphe 1.1.) inhérents à toute documentation manuelle.

Nous terminons le tour d'horizon concernant la documentation des applications en considérant les dictionnaires gérés par les systèmes.

Celui qui nous intéresse dans le cadre de l'UCM est le dictionnaire du SGBD SQL.

### 2.3.2. LES TABLES DU SYSTEME SQL.

Les fichiers de type 'BD' sont implémentés en SQL. Le SGBD SQL tient à jour des informations concernant la base de données dans un ensemble de tables appelées 'catalogues'. Ces catalogues sont créés automatiquement lors de la génération de la base de données et décrivent les objets du système utilisateur gérés par SQL; ils constituent le dictionnaire SQL de données.

Les catalogues sont tenus à jour automatiquement lors de l'exécution des requêtes SQL. Ils sont consultables par l'utilisateur comme n'importe quelle autre table d'accès public.

Afin de se faire une idée plus précise du contenu de ce dictionnaire, il nous a semblé utile de reconstituer à l'aide du schéma d'implémentation (tables système SQL), le schéma conceptuel sous-jacent (cf fig.8).
Notons que ce schéma n'est pas complet et que la liste des attributs qui en est donné n'est pas exhaustive.

2.3.2.1.

Toute occurrence de SYSCATALOG décrit soit une table, soit une vue de la B.D., ainsi que les catalogues eux-memes.

Attributs:

- TNAME: nom de la table ou de la vue.
- CREATOR: identifiant du créateur de la table/vue; = 'SYSTEM', s'il s'agit de catalogues.
- REMARkS: texte descriptif de la table/vue (254 caratères)
  (...)

- Identifiant:

- TNAME, CREATOR.

2.3.2.2.

Toute occurrence de SYSCOLUMNS décrit chaque colonne de chaque table ou vue de la B.D. (catalogues inclus)

Attributs:

- CNAME: nom de la colonne.
- COLNO: numéro de la colonne dans la table/vue.
- COLTYPE: type de valeur associé à la colonne.
- LENGTH: longueur associée au type de valeur.
- NULLS = 'Y' si absence de valeur autorisée.
- REMARKS: texte descriptif de la colonne (254 caractères maximum)
  (...)

Identifiant:

CNAME, id (SYSCATALOG).

2.3.2.3.

Toute occurrence de SYSINDEXES décrit chaque index courant existant dans la B.D. ainsi que ceux utilisés par SQL/DS sur ses propres catalogues.

Attributs:

- INAME: nom de l'index.
- ICREATOR: identifiant du créateur de l'index.
- INDEXTYPE = 'U' si index unique.
  = 'D' si doubles autorisés.

(...)

Identifiant:

INAME, ICREATOR.

2.3.2.4.

Toute occurrence de SYSDBSPACES décrit chaque DBSPACE de la B.D. y compris des DBSPACES non assignés.

Attributs:

- DBSPACENO: numéro du DBSPACE.
- DBSPACENAME: nom de l'utilisateur du DBSPACE.
- OWNER = '  ' si DBSPACE non assigné.
  = identifiant si DBSPACE privée (=mot de passe).
  = 'PUBLIC' si DBSPACE publique.
- NTABS: nombre de tables contenues dans la DBSPACE.
- NPAGES: nombre de pages utilisables.

(...)

Identifiant:

DBSPACENO.

2.3.2.5.

Toute occurrence de SYSVIEWS décrit chaque vue connue du système sous la forme de lignes d'instructions SQL définissant les vues (VIEWTEXT).

Attributs:

- SEQNO: Une instruction SQL pouvant contenir plus de 254 caractères, on définit différentes lignes d'instruction ordonnées sur SEQNO.
- VIEWTEXT: contient la requete/partie de requete SQL définissant la vue.

Identifiant:

SEQNO, id (SYSCATALOG).

2.3.2.6.

Toute occurrence de SYSTABAUTH décrit les privilèges d'accès aux tables/vues détenus par les utilisateurs ainsi que la source du privilège.

Toute occurrence décrit également les privilèges sur les tables/vues exercés par divers programmes préprocessés (modules d'accès).

Attributs:

- GRANTOR: identifiant du donneur des privilèges.
- GRANTEE: identifiant du bénéficiare des privilèges.
  = 'PUBLIC' si tous les utilisateurs sont bénéficiaires du privilège.
- GRANTEETYPE = ' ' si le bénéficiare est un utilisateur.
  = 'p' si le bénéficiare est un programme.
  Dans ce cas, GRANTOR est identifiant du programme correspondantau module d'accès, GRANTEE est le nom du programme bénéficiaire.
- SELECTAUTH = 'Y' si l'utilisateur est autorisé à lire des lignes de la table/vue cible.
  = 'G' si le privilège de lecture est transmissible.
  = ' ' dans les autres cas.

- INSERTAUTH = 'Y' si l'utilisateur est autorisé à écrire des lignes de la table/vue cible.
  = 'G' si le privilège d'insertion est transmissible.
  = ' ' dans les autres cas.
- UPDATEAUTH = 'Y' si mise à jour autorisée.
  = 'G' si privilège de mise à jour transmissible.
  = ' ' dans les autres cas.
- DELETEAUTH = 'Y' si l'utilisateur est autorisé à effacer des lignes de la table/vue cible.
  = 'G' si privilège d'effacement transmissible.
  = ' ' dans les autres cas.
- ALTERAUTH = 'Y' si l'objet est une table de base et que l'utilisateur est autorisé à l'étendre
  = 'G' si le privilège d'extension est transmissible.
  = ' ' dans les autres cas.

Identifiant:

GRANTOR, GRANTEE,id (SYSCATALOG via SOURCE), id (SYSCATALOG via TARGET).

2.3.2.7.

Toute occurrence de SYSACCESS décrit les modules d'accès qui ont été créés pour des programmes utilisateurs par le préprocesseur SQL/DS.

2.3.2.8.

Toute occurrence de OBJET représente une occurence d'un des types suivants:

Attribut:

- TYPE = 'R': l'objet est de type SYSCATALOG.
  'I': l'objet est de type SYSINDEX.
  'S': l'objet est de type SYSDBSPACE.
  'X': l'objet est de type SYSACCESS.

Identifiant:

- TYPE, id (type auquel se rapporte l'objet)

2.3.2.9.

Toute occurrence de SYSUSAGE décrit les relations de dépendance entre un objet de base et un objet dépendant.
Ainsi, une vue dépend des tables de même qu'un module d'accès dépend de tables et d'indexes.

2.3.2.10.

Toute occurrence de SYSCOLAUTH décrit les privilèges d'accès définis sur une colonne d'une table/vue cible.

2.3.2.11.

Toute occurrence de SOURCE décrit la table/vue source sur laquelle sont donnés des privilèges.

2.3.2.12.

Toute occurrence de TARGET décrit la table/vue cible sur laquelle le bénéficiaire possède des privilèges.

2.3.2.13.Contraintes d'intégrité.
---

- Une occurrence d'objet ne peut etre à la fois origine et cible d'une association SYSUSAGE.

- Si une occurrence de SYSINDEXES est associée à plusieurs occurrences de SYSCOLUMNS, ces dernières sont associés à une meme occurrence de SYSCATALOG de type = 'R' (un index est défini sur une seule table).

- Une occurrence de SYSTABAUTH définissant deux occurrence de SYSCATALOG (une via SOURCE et une via TARGET), ces deux occurrences sont:
- soit identiques et représentent des tables (pas d'association avec SYSVIEWS).
- soit distinctes et l'une (SOURCE) représente une table tandis que l'autre (TARGET) représente une vue définie sur la dite table (mapping SYSUSAGE entre table et vue correspondante).

- Si une occurrence de SYSTABAUTH est associée à plusieurs occurrences de SYSCOLUMNS, ces dernières sont associéS à une meme occurrence de syscatalog, laquelle est associée via TARGET à l'occurrence SYSTABAUTH de départ (un privilège est défini sur une seule table ou vue).

Les catalogues SQL constituent un mini dictionnaire implémenté en tables SQL et décrivant l'implémentation des schémas SQL de la B.D. à l'aide
-d'objets traduisant les concepts de tables, vues, dbspaces, privilèges, colonnes, index, module d'accès
-de propriétés spécifiques aux objets
-de relations spécifiques inter-objets.

Ils représentent un outil de documentation facilement exploitable et automatiquement mis à jour. Cependant,ils ne concernent que les données impléméntées en SQL et les modules d'accès y afférant. On n'y trouvera donc aucun renseignement concernant
les schémas -autres que SQL
-autres que d'implémentation
les traitements -autres que ceux définissant des modules d'accès à la BD SQL

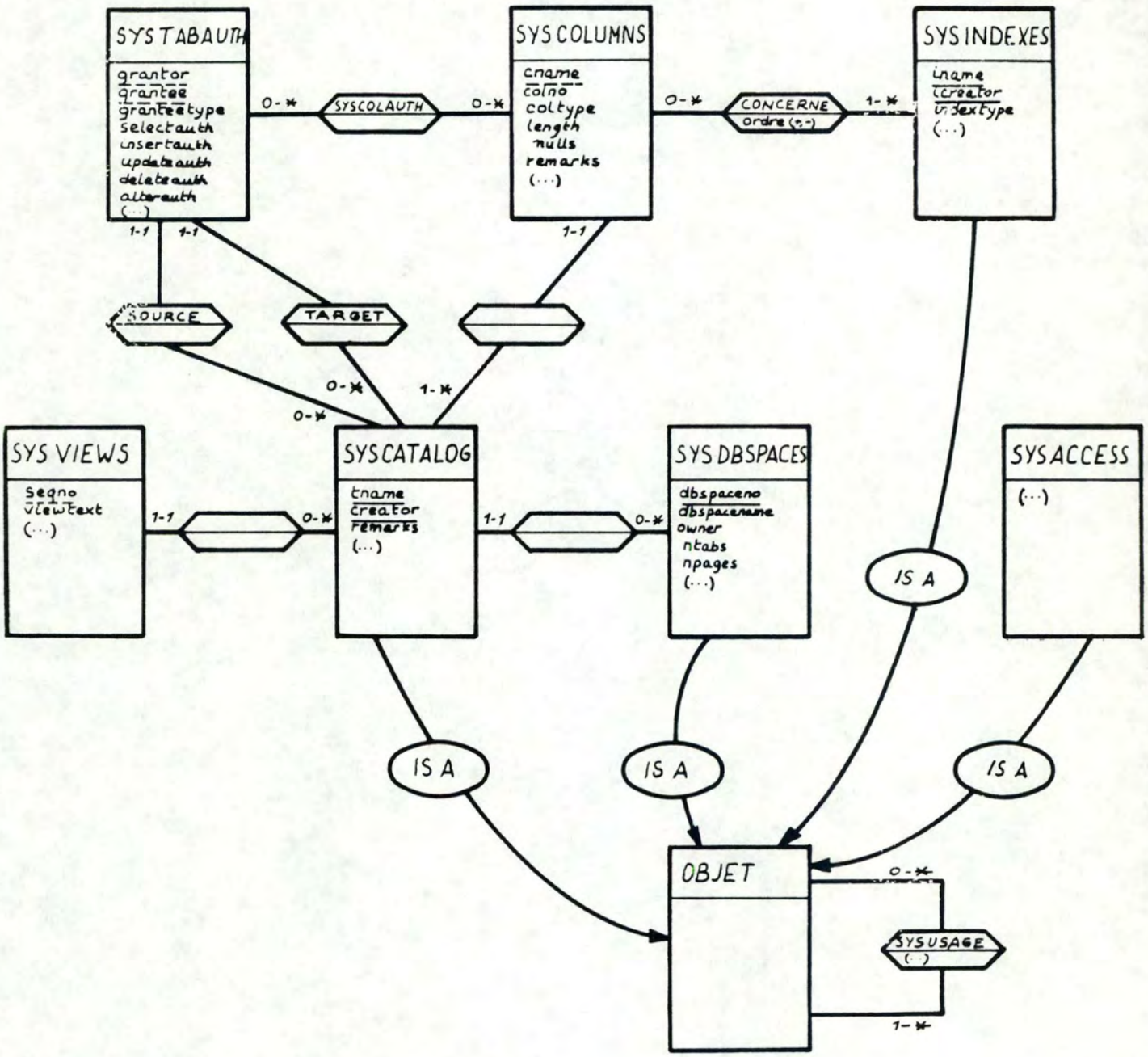

SYS TABAUTH
grantor
grantee
granteetype
selectauth
insertauth
updateauth
deleteauth
alterauth
(...)
0-*
SYSCOLAUTH
0-*
SYS COLUMNS
cname
colno
coltype
length
nulls
remarks
(...)
0-*
CONCERNE
ordre (+,-)
1-*
SYS INDEXES
iname
icreator
indextype
(...)
1-1
1-1
1-1
SOURCE
TARGET
0-*
0-*
1-*
SYS VIEWS
seqno
viewtext
(...)
1-1
0-*
SYSCATALOG
tname
creator
remarks
(...)
1-1
0-*
SYS DBSPACES
dbspaceno
dbspacename
owner
ntabs
npages
(...)
SYSACCESS
(...)
IS A
IS A
IS A
IS A
OBJET
0-*
SYSUSAGE
(...)
1-*

figure 8

# CHAPITRE 3

# CADRE DU PROJET

## 3.1. BESOINS EN DOCUMENTATION

Au vu des outils documentaires d'usage à l'UCM, il apparait clairement les besoins d'une documentation :

- plus riche
  couvrant tous les niveaux de description correspondant aux différentes étapes du cycle de vie du SI et permettant de décrire des correspondances entre méta-données en vue de rendre possible la gestion de la ressource information;

- plus facilement exploitable
  tant au point de vue de la maintenance que de l'interrogation et de la production de rapports d'analyse et de synthèse en vue de rendre effective la gestion de la ressource information.

L'outil SDD parait dès lors tout indiqué pour répondre à ces besoins.

## 3.2. CADRE DU PROJET

L'objectif de ce travail consiste à montrer la puissance potentielle de l'outil SDD et plus particulièrement l'adéquation de cet outil aux besoins documentaires de l'UCM.

Par conséquent, nous développerons un SDD passif dont le DD sera géré par le SGBD SQL existant (dépendance forte).

Dans la suite de l'exposé, nous tenterons de définir un modèle autant que possible indépendant d'une méthode ainsi que quelques fonctions d'utilisation.

### 3.2.1.Modèle

L'analyse conceptuelle tentera de définir un modèle prenant en compte tous les aspects de description d'un SI.

Celui-ci integrera notamment les modèles
- de structuration des données
- de structuration et dynamique des traitements
- de statique des traitements
- des ressources

décrits dans (11).

Il permettra en outre de définir des correspondances (ou "mapping") entre groupes d'occurrences appartenant à un même concept ou à des concepts différents du modèle.

De plus, possibilité sera donnée d'associer une ou plusieurs lignes de texte à toute occurrence d'un concept du modele

Enfin,le modèle sera extensible ou indépendant vis à vis d'une organisation.

Le modèle spécifié sera de type E.A.

### 3.2.2 Outils.

Nous utiliserons les outils qu'offre le SGBD SQL en vue
- d'implémenter le dictionnaire de données
- d'en assurer l'exploitation.

#### 3.2.2.1. Fonctions de documentation

Les fonctions de documentation (opérations de création, mise à jour, suppression des méta-données et interrogation du dictionnaire) seront assurées à l'aide de
- requêtes SQL simples
- requêtes SQL enregistrées, paramétrées
- "run" SQL

#### 3.2.2.2.Fonctions de contrôle de cohérence et validité.

En l'absence d'outil de contrôle existant , nous spécifierons un programme de chargement assurant certains contrôles de validité interne.

#### 3.2.2.3. Fonctions de conversion.

Nous verrons dans quelle mesure il est possible d'automatiser le transfert des méta-données du SGBD SQL dans le dictionnaire qu'il gère.

### 3.2.3. Méthode

Celle-ci propose une organisation de l'activité de conception et de développement des applications dans le but d'en contrôler les différentes étapes standardisées et d'améliorer les performances et la rentabilité.

Cependant, afin de garantir la portabilité et la souplesse du dictionnaire, nous spécifions celui-ci indépendamment de toute méthode.

Il reste que le dictionnaire devrait pouvoir prendre en compte et intégrer les diverses méthodes de conception et de développement de logiciels proposées.

## 3.3. QUELQUES TYPES DE REQUETES UTILES

Nous retiendrons quelques types de requêtes générales, caractérisant l'exploitation du dictionnaire.

Loin d'être exhaustives, elles illustrent dans un formalisme libre et indépendant de toute méthodologie précise certaines catégories de besoins que le SDD veillera à satisfaire.

1) Etant donné un traitement (à un niveau de développement quelconque),

- quelles en sont les sous-structures particulières?
- quelles sont les structures de données manipulées et leurs modalités de manipulation (lecture, écriture, consultation, mise à jour ...)?

2) Etant donné un traitement (à un niveau de développement quelconque),
- quel(s) est (sont) le(s) traitement(s) correspondant(s) au niveau de développement suivant?

3) Etant donné un traitement,
- de quelles ressources "hard /soft" a-t-il besoin?
- en combien de versions est-il stocké physiquement?
  pour chaque version: - nom physique
  - unité de stockage
  - raison d'être

4) Etant donné un traitement en exploitation,
- quels fichiers manipule-t-il?
- modalité de manipulation
- description du traitement

5) Etant donné un service (département),
-quels sont les traitements qu'il définit ainsi que le niveau de développement de ceux-ci?

6) Etant donné un traitement en développement,
- qui en est le responsable?
- quelles en sont les spécifications?
- quelles sont les différentes étapes déjà réalisées?
- que reste-t-il à faire?

7) Etant donné un traitement en conversion,
- quelle est la raison de la conversion?
- quel est le traitement originel?

8) Etant donné une structure (sous-structure) de données,
- quelle est la structure (sous-structure) de données correspondant au niveau de développement suivant?

9) Etant donné une structure de données (à un niveau de développement quelconque),
- quelles en sont les sous-structures particulières correspondant à un critère donné (utilisateur ou traitement)?

10) Etant donné une structure de données (à un niveau de développement quelconque),
- donner la description de cette structure.

11) Etant donné un utilisateur,
- quelle est sa machine virtuelle?
- quels sont ses fichiers?
- quel est son DBSPACE?

12) Etant donné un fichier,
- quel est son type (base, mouvement, historique, archive,...)?
- sur quelle unité de mémoire secondaire est-il stocké?
- quel est son nom physique?

13) Etant donné une vue SQL,
- sur quelle(s) table(s) est-elle définie?
- quelles sont ses caractéristiques?

# CHAPITRE 4

# CONCEPTION DU DD

## 4.1. STRUCTURATION DES DONNEES

### 4.1.1.Notions liées à l'implémentation

En informatique, la nécessité d'organiser les données est un problème connu depuis assez longtemps. De nombreux modèles de description ont été développés. Cependant, un grand nombre d'entre eux se limite à la dernière étape du développement d'une structure de données : son implémentation. De plus, les concepts développés sont souvent liés à un système de gestion particulier, qu'il soit peu évolué comme un ensemble de fichiers COBOL ou qu'il soit un SGBD plus moderne comme SQL ou CODASYL.

Quels sont ces concepts ?
Nous allons tenter de répondre à cette question en trois approches :

- Premièrement, nous avons repris certains éléments d'une étude assez ancienne sur les dictionnaires de données (X) qui en comparait quatre d'entre eux : "Datamanager", "British Rail", "Remora", "Uhrowczik".

- Ensuite, nous reprendrons les notions principales liées aux données qui se retrouvent dans le dossier informatique de la CAS développé à l'UCM et qui se sont détachées de nos entretiens avec Monsieur Saint Amant, directeur du service informatique.

- Finalement, nous développerons les éléments généraux du dictionnaire de données lié au SGBD SQL utilisé à l'UCM.

#### 4.1.1.1.Synthèse de quatre dictionnaires de données

Remarque :

Le lecteur pourra s'étonner du caractère peu formel et peu précis des définitions qui vont suivre. Celles-ci mélangent des notions très différentes sur l'organisation physique du stockage en mémoire secondaire alors que celle-ci est par définition variable. Cette approche a pour but principal d'énumérer un premier ensemble de notions que nous devons modéliser et de montrer la nécessité du développement d'un modèle à la fois plus large et plus fin.

Comme la terminologie utilisée est non standardisée, nous donnerons pour chaque notion le terme qui nous parait le plus clair ainsi qu'un ensemble de synonymes.

1) LA DONNEE ELEMENTAIRE=(ELEMENT = CHAMP = MOT).

C'est la plus petite unité d'information, un atome élémentaire (ce sera par exemple, le nom d'un employé, sa date de naissance, son sexe,...). Ses propriétes reprennent généralement :

- une description (formelle ou informelle)
- le nom et/ou le numéro
- le type (numérique, alphabétique ...)
- la longueur (en caractères, en"bits")
- la version et la date
- la valeur par défaut ou la possibilité de valeur nulle
- les caractéristiques d'édition (format, alignement, règles,...)
- des caractéristiques propres au rôle joué dans une structure plus importante:
    - si elle est une clé
    - si elle est un index
    - sa position séquentielle

## 2) LE GROUPE (= SEGMENTS LOGIQUES = ENREGISTREMENTS LOGIQUES)

C'est une première étape de structuration: le GROUPE est un ensemble de DONNEES ELEMENTAIRES et/ou d'autres GROUPES qui sont alors construits récursivement. Ses propriétés sont similaires à celles de l'élément ce qui peut:

- traduire une simple redondance
  (ex: longueur du groupe = la somme des longueurs des éléments)
- introduire une confusion
  ( ex: chaque élément est-il une clé ou le groupe entier forme-t-il une seule clé ?)

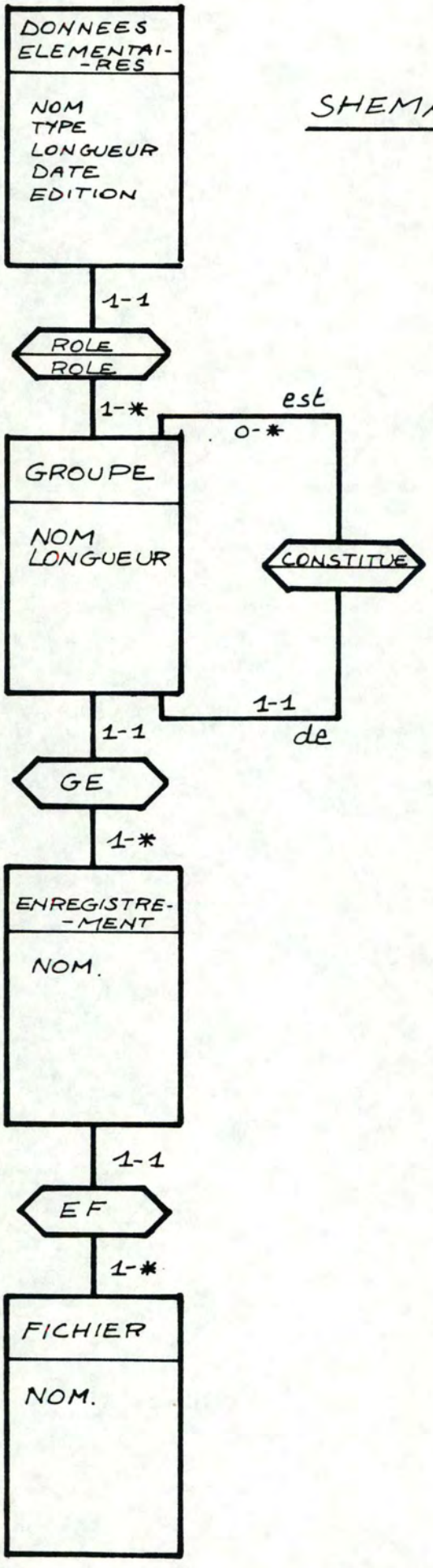
## 3) L'ENREGISTREMENT (= SEGMENT PHYSIQUE = "RECORD".)

Cette notion est liée à l'accès. C'est la plus petite unité référencée par un traitement sur la mémoire secondaire en lecture ou en écriture. Un ENREGISTREMENT peut s'identifier à un ou plusieurs groupes; dans ce cas, une seconde étape de structuration est réalisée. Ses propriétés se limitent généralement au nom et à la longueur.

## 4) LE FICHIER.

Notion bien connue des programmeurs, le FICHIER est un ensemble d'ENREGISTREMENTS, non nécessairement du même type et référencé par un nom unique. On ouvre un FICHIER en début de traitement et on le ferme en fin

# SHEMA CONCEPTUEL



Evaluation
---

Le schéma ainsi défini présente un FICHIER comme unique. Nous verrons que ce concept ainsi défini est insuffisant.

Les notions mêmes de FICHIER et de SEGMENTS PHYSIQUES ne sont plus d'actualité dans les SGBD modernes qui implémentent directement des concepts de plus haut niveau (cfr autres approches).

Une confusion apparait entre la DONNEE ELEMENTAIRE et le GROUPE car ces notions sont mal définies. Bien qu'une structure complexe peut être représentée, il est possible que deux utilisateurs décrivent différemment un même FICHIER, le premier considérant un niveau de regroupement supérieur au second.

L'identifiant d'un enregistrement n'est pas précisé.

Les notions d'ordre d'accès entre ENREGISTREMENTS et de clé de tri des FICHIERS ne sont pas représentés car les clés et indexes définis ne sont pas suffisamment précis.

4.1.1.2.Le dossier informatique de la CAS
---

En limitant notre étude de ce dossier aux informations liées aux données, nous pouvons repérer de nouvelles notions:
- le mode d'organisation des fichiers, que nous appellerons le type; l'UCM emploie des initiales : DBS, KSDS, (b)CMS, (b)DB (cfr chapitre 3)
- un même fichier peut exister en différents exemplaires suivant le rôle qu'il joue dans l'organisation: sécurité (Back-up), traitements ultérieurs (versions, historique), besoins d'archivage.

De ceci découle que pour affiner la notion de FICHIER introduite précédemment, nous avons plusieurs possibilités.

Nous pouvons lui conserver sa définition de regroupement de segments physiques sur mémoire secondaire, c'est-à-dire que chaque exemplaire effectivement enregistré correspondra à une occurrence du type. Nous traduirons le rôle que jouent certains de ces FICHIERS par rapport à d'autres en ajoutant un T.A récursif sur le T.E : le T.A ROLE, avec un attribut précisant si c'est un back-up, une version ou un archivage. Le type du fichier est un attribut supplémentaire du T.E.

Cette solution a l'avantage de faire l'économie de plusieurs nouveaux T.E. répartissant les fichiers suivant leur rôle. Par contre, nous introduisons une lourde redondance car pour être complets, nous devons décrire complétement chaque fichier, c'est-à-dire avec tous ses enregistrements, groupes et éléments, alors que certains de ceux-ci seront rigoureusement identiques pour différentes occurrences de FICHIER.

Une meilleure solution est d'introduire un nouveau type généralisant la description de plusieurs fichiers physiquement implantés lorsque celle-ci leur est strictement commune. Remarquons que par ce raisonnement, nous nous écartons des concepts liés à la seule implémentation pour nous situer à un niveau d'abstraction suppérieur. Nous conservons le fichier physique comme notion directement liée à l'enregistrement en mémoire, il constitue un autre T.E ou plutôt un T.A que nous appellerons localisation du fichier (LOCF). Un même fichier logique (FICHIER) peut en posséder plusieurs, chacune jouant un rôle particulier (attribut du T.A)

Nous ne préciserons pas à quel autre T.E est relié LOCF à ce niveau, nous développons par ailleurs les différentes ressources de l'organisation.

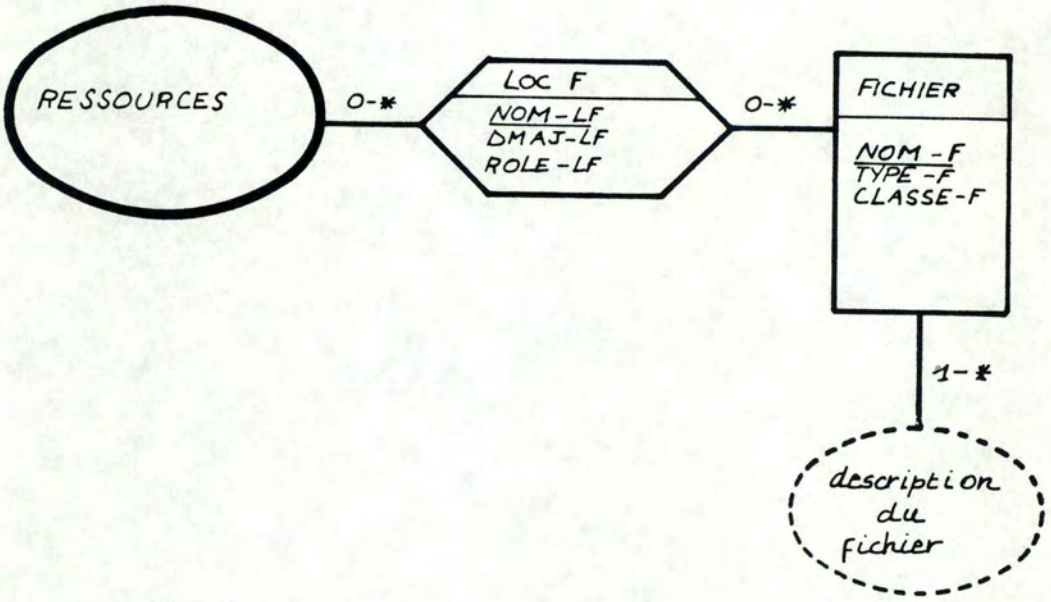
Une autre remarque concerne la date de dernière mise à jour d'une DONNEE ELEMENTAIRE. Nous l'associerons plutôt à chaque localisation d'un fichier, d'une part parce que la description ainsi définie est essentiellement statique et d'autre part parce que cette précision pour chaque élément ne nous est pas utile.

Remarquons pour terminer que certains fichiers ne feront l'objet d'aucune localisation physique : ce sont les fichiers temporaires créés et détruits par un même traitement.

Nous n'imposerons donc pas aux fichiers la contrainte d'être l'objet d'au moins une localisation et nous ajouterons un attribut au T.E : la classe, qui précise s'il est permanent ou temporaire.

Nous identifierons chaque fichier par son nom ainsi que chacune de ses localisations physiques.

Nous obtenons le schéma suivant :




Evaluation
---

La structure (FICHIER + LOCF) apparait déjà mieux définie et nous semble moins restrictive car elle va nous permettre de traduire des concepts plus larges que ceux définis pour les seuls fichiers conventionnels (point suivant). Par ailleurs, la méthode de description du fichier n'a pas été modifiée et conserve tous ses inconvénients.

#### 4.1.1.3.Les tables SQL

Le dictionnaire de données du SGBD SQL regroupe plusieurs sortes de descriptions; outre les données, il nous informe également sur la méthode de partage des ressources (SYSDBSPACES) que nous ne considérons pas ici.

D'une part, nous avons pour but de construire un modèle général c'est-à-dire non lié à un modèle de données particulier comme le relationnel qui est sous-jacent au SGBD SQL de l'IBM (cfr annexe 5.1 pour plus de détail sur le relationnel). Mais d'autre part, notre conception est liée à ce SGBD car une grande partie des traitements informatiques de l'UCM est réalisée en SQL et celle-ci doit s'accroître dans l'avenir.

1) Les tables et les vues

La table SYSCATALOG décrivant les tables et vues de la BD peut être considérée comme un sous-ensemble du T.E.: FICHIER. En effet, celui-ci est un regroupement de définitions au même titre qu'une relation (modèle relationnel). Cette correspondance nous a déjà été suggérée à l'étape précédente car dans le dossier informatique (UCM) les fichiers peuvent être de type 'BD' c'est-à-dire une table ou une vue SQL.

Une restriction doit cependant être faite pour les vues. Celles-ci correspondent à un filtre sur une ou plusieurs tables, ou plus précisément sur certaines colonnes de certaines tables. Celles-ci ont donc déjà été décrites pour les tables.

Plus généralement, ceci nous amène à considérer qu'un ensemble de définitions peut être partagé par plusieurs fichiers, indépendamment de la localisation physique de ceux-ci. Par ailleurs, la notion de colonne n'est pas représentée clairement par l'un de nos T.E.: SEGMENT PHYSIQUE, GROUPE ou DONNEE ELEMENTAIRE. Le besoin d'un modèle général regroupant l'ensemble des définitions de données est maintenant clairement établi.

On peut également remarquer qu'une vue ne fait l'objet d'aucun stockage en mémoire secondaire. En effet, la requête de création d'une vue ne suppose qu'une sélection et une présentation de données déjà enregistrées.

2) Les indexes

Dans une table ou une vue SQL, la notion de clé d'accès n'a pas de sens. En effet, une requête peut définir des contraintes sur une ou plusieurs colonnes d'une table ou même de plusieurs tables (requetes imbriquées). La définition d'indexes permet d'une part d'améliorer les performances et d'autre part, en utilisant la clause "unique index", définit un identifiant (formé lui aussi sur une ou plusieurs colonnes).

La table SYSINDEX cache donc deux notions d'importance différente : la seconde, l'identifiant, est très importante dans la définition d'un modèle de données, la première, par contre, est uniquement liée aux performances.

3) Le créateur et les privilèges

Le SQL associe un créateur à chaque table ou plutôt une machine virtuelle, notion du système d'exploitation chez IBM (définie par ailleurs). Chaque créateur peut également définir des privilèges d'accès pour chacune des tables qu'il crée et même pour chaque colonne. Ceux-ci sont accordés à d'autres utilisateurs ou a des modules d'accès.

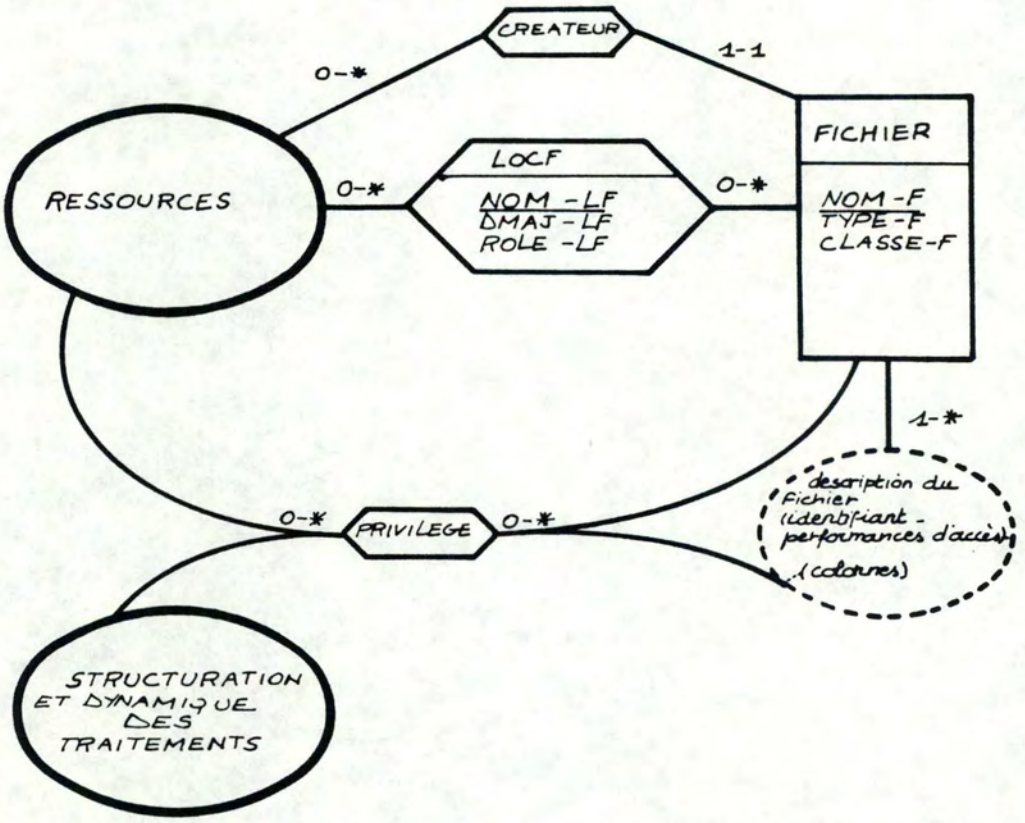
Ne nous intéressant qu'aux données, nous devons malgré tout introduire ces notions comme des liens avec d'autres sous-modèles :- RESSOURCES
- DYNAMIQUE et STRUCTURATION DES TRAITEMENTS.

4) Les relations de dépendances

La table SYSUSAGES définit les relations de dépendances dont nous avons déjà parlé, distinguons-les:

- vues - tables
- indexes - tables (ou vues)
- tables (ou vues) - dbspaces
- modules d'accès - tables (ou vues)

Schéma conceptuel.



### 4.1.2 Un modèle conceptuel

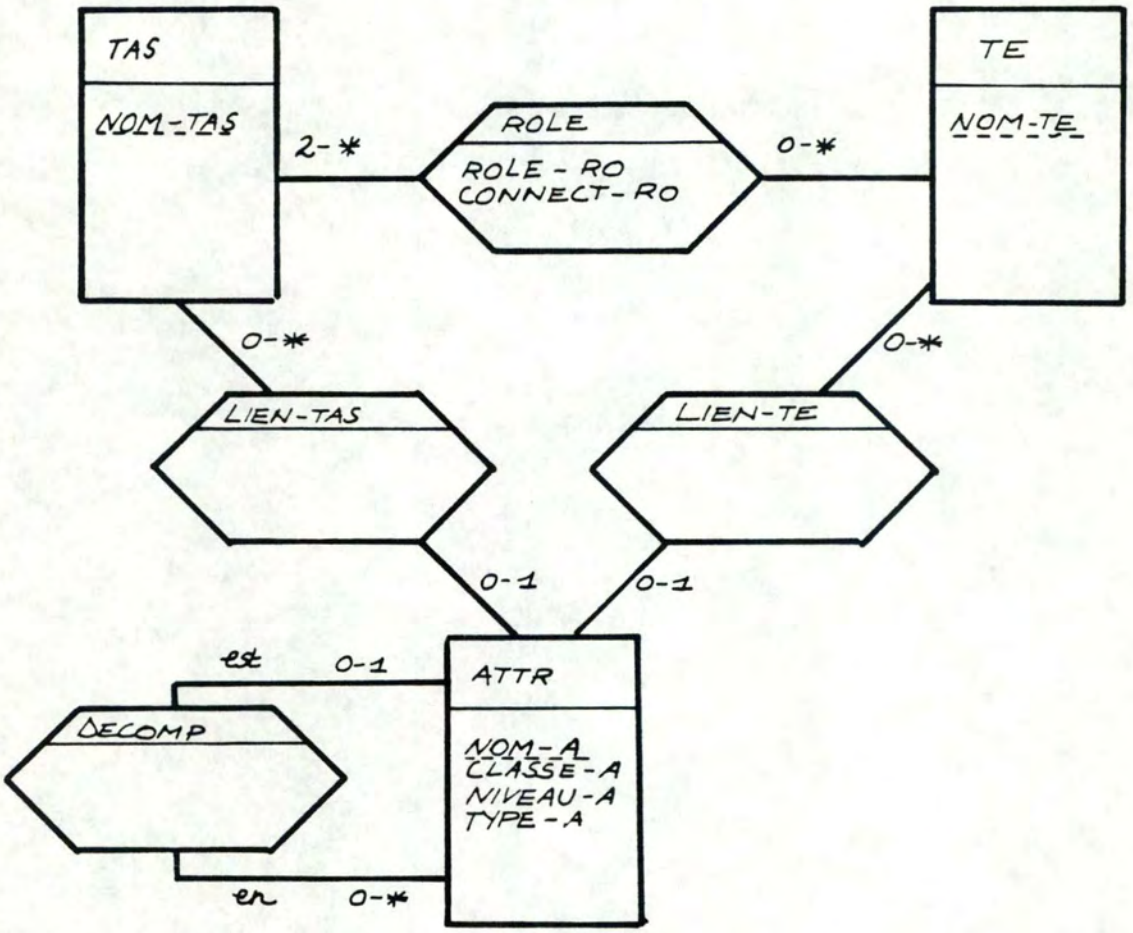
Les composants définis jusqu'à présent ne nous ont pas permis de modéliser de façon satisfaisante la structure des données proprement dite. Nous savons que celle-ci peut-être commune à plusieurs fichiers et inversément qu'un fichier peut en contenir plusieurs. Notre problème est lié à l'approche par l'implémentation qui ne peut mettre en commun des concepts particuliers. De plus, notre dictionnaire a pour but de décrire toutes les étapes du développement dont la première est la conception.

Le modèle conceptuel de structuration des informations dont l'utilisation tend de plus en plus à se généraliser est le modèle E.A. Comme notre but est la description de données, nous allons modéliser le formalisme E.A.

Nous limiterons cet objectif ambitieux à la représentation du T.E, du T.A, de l'attribut et des quelques notions principales relatives à chacun de ceux-ci (l'annexe 5.1 définit ces concepts).

Distinguons bien cette démarche de la présentation qui en est faite, dans notre cas, nous utiliserons également le formalisme E.A pour présenter le schéma du dictionnaire.

Schéma conceptuel



Identifiants :
---

Chaque T.E., TAS possède un nom qui l'identifie dans un schéma conceptuel particulier. Cette dernière notion n'a pas encore été introduite dans notre molèle. Supposons provisoirement un seul schéma dans lequel NOM-TE et NOM-TAS sont des identifiants propres.

Pour l'attribut, adoptons la convention que son nom l'identifiera des autres pour un T.E. ou un TAS particulier; nous obtenons : id (ATTR) = NOM-A + id(T.E. ou TAS)

T.A. ROLE
---

Chaque T.E. participant à un T.A. peut y jouer un rôle particulier, ce lien est également caractérisé par une connectivité. Remarquons que le modèle peut traduire un T.A. récursif ou binaire, simple, ternaire...

Contrainte d'exclusion entre LIEN-TE et LIEN-TAS
---

Chaque attribut doit etre relié obligatoirement à un T.E. ou à un TAS, mais il ne peut être lié qu'à un de ceux-ci.

T.A. DECOMP
---

Si un attribut est décomposable, il sera relié à tous les attributs qui le composent, le graphe obtenu formant une arborescence; ceux-ci restent cependant reliés à leur T.E. ou TAS ce qui introduit une contrainte que nous aurons à gérer (redondance).

T.E. ATTR
---

Les propriétés générales d'un attribut (obligatoire ou facultatif, élémentaire ou décomposable, simple ou répétitif) sont représentées par la classe qui reprend, dans cet ordre, la première lettre de chaque mot (ex : OES, FDR,...). La seconde propriété introduit une contrainte sur le T.A. DECOMP, ainsi que la troisième, NIVEAU-A qui précise le numéro de niveau dans l'arborescence. Toutes ces répétitions se justifieront mieux dans la suite.

Le type définit le type de valeur de l'attribut ainsi que son format (nombre de caractères, emplacement du signe, de l'exposant,...)

Autres notions
---

La possibilité reste offerte à l'utilisateur d'associer une ou plusieurs lignes de texte à chaque occurrence de chaque type défini ( cfr chap. 3 pour le principe général de l'association du T.E. TEXTE avec tout autre T.E ; cfr chap.5 pour l'implémentation). Ce mécanisme sera par exemple utile pour :

- la définition d'un T.E., TAS ou ATTR,
- la cardinalité d'un T.E. ou TAS,
- les valeurs d'un attribut (ensemble de valeurs possibles, valeur par défaut,...)
- les dépendances fonctionnelles,
- d'autres contraintes d'intégrité.

Evaluation
---

Ce modèle très riche nous permet de représenter sous un formalisme standardisé une organisation de données. Nous pouvons les associer conceptuellement, ce qui était impossible avec nos fichiers indépendants les uns des autres. Ce modèle est généralement reconnu comme un point de départ qui peut aboutir à une implémentation dans un environnement quelconque, évolué ou non, qui est le point d'arrivée.

Certaines notions propres à la correspondance entre ces deux points ne sont pas encore bien représentées, comme la clé d'accès ou l'ordre d'accès. Nous devons préciser les liens entre ce modèle et le T.E. fichier défini précédemment. Nous devons également développer la notion de schéma (conceptuel ou autre) qui distingue une structure d'une autre, notion qui ne pouvait être introduite à ce niveau car le modèle E.-A. représente lui-même un schéma.

### 4.1.3.Une liaison entre la conception et l'implémentation

#### 4.1.3.1.Pour conserver le formalisme E.A.

La théorie de la conception de bases de données est un sujet largement développé dans la littérature et celle-ci dépasse le cadre de ce travail. Nous nous limiterons à citer les grands principes qui nous ont amenés à développer notre modèle. Ceux-ci sont liés à une méthodologie enseignée aux FNDP (Namur) mais nous espérons rester à un niveau suffisamment général pour laisser à l'utilisateur le choix de sa méthode.

La conception d'une structure de données est décomposable en phases, dont la première est l'analyse conceptuelle et la dernière est l'implémentation. A chacune de celles-ci, nous associons un schéma traduisant les décisions prises successivement par le concepteur. Il est clair que ces différents schémas sont reliés entre eux : une correspondance doit être faite entre une sous-structure d'un schéma et celle d'un autre. Par ailleurs, les différents schémas peuvent utiliser des formalismes différents qui sont traduits de l'un à l'autre.

Dans notre dictionnaire, nous avons conservé la terminologie du modèle E.A. pour toutes les étapes du développement que l'utilisateur désirera décrire. Ceci se justifie par notre souci de généralité et par la facile traduction de ce formalisme en d'autres.

Distinguons bien cette traduction de la correspondance introduite précédemment sur un exemple classique. Nous utilisons le modèle MAG comme second formalisme (l'annexe 5.1 définit brièvement ce modèle et l'annexe 5.2 donne un exemple plus détaillé de la correspondance).

Modèle E.A.
---
- types d'entité : CLIENT, COMMANDE
- types d'associations : PASSATION
- attributs : NOM-CLI, NOM-COM, DATE-PASSATION

Correspondance MAG
---
- types d'articles : CLIENT, COMMANDE, PASSATION
- types de chemins : CLI-PAS, COM-PAS
- items : NOM-CLI, NOM-COM, DATE-PASSATION

Traduction du schèma MAG obtenu dans le formalisme E.A.
---
- types d'entités : CLIENT, COMMANDE, PASSATION
- types d'associations : CLI-PAS, COM-PAS
- attributs : NOM-CLI, NOM-COM, DATE-PASSATION

Grace à cette restriction, nous pouvons conserver les notions définies au point précédent pour toutes les étapes de la conception : T.E., TAS, ATTR et les T.A. qui les relient.

#### 4.1.3.2.Le T.E.: SCHEMA

Le SCHEMA regroupe un ensemble de définitions d'une structure de données à une étape quelconque de sa conception. Il est identifié par son nom. Il est caractérisé par son type (description informelle suivant le choix de la méthode), son stade de développement (terminé ou non), son numéro de version (si plusieurs sont conservées) et sa date de dernière mise à jour. Nous le relions à T.E. car chaque TAS est défini sur un ou plusieurs T.E. et chaque ATTR est relié à un T.E. ou un TAS.

Les privilèges et le créateur que nous avions définis sur le fichier ou une partie de la description de ce fichier seront associés directement au SCHEMA. Nous allons voir comment le fichier est lui-même associé à cette description. Il est clair qu'une permission d'accès est plutôt garantie sur une structure de données. Si, exceptionnellement, ce privilège se limite à un fichier ou une partie de fichier, il est nécessaire de créer un schéma réduit à la portée de celui-ci. Ceci nous permet d'ajouter un attribut au schéma précisant si un ou plusieurs privilèges d'accès sont accordés ou s'il est d'accès public. Pour limiter le nombre de liens entre les différents éléments, nous regroupons les deux notions de privilège et de création en un seul T.A. dont le rôle précise, pour chaque occurrence, le type de lien établi.

Le mécanisme des MAPPING défini précédemment nous permet d'établir très précisément la correspondance entre les sous-structures de schémas différents.

#### 4.1.3.3.La liaison FICHIER - description d'une structure

Une structure quelconque définie par un schéma ou une partie de celui-ci, peut être regroupée logiquement dans un FICHIER, ce qui en précise la signification. Celui-ci peut alors être l'objet de différentes localisations physiques comme nous l'avons vu. L'élément de base d'une structure est le T.E. comme nous l'avons établi pour la liaison vers le schéma.

Nous relions également le T.E. au FICHIER en précisant que celui-ci peut en regrouper plusieurs (différents types de"records" dans un fichier COBOL par exemple) mais en contient au moins un. Inversément, il n'est pas obligatoire d'organiser la découpe en fichiers surtout dans les premières étapes de la conception, c'est-à-dire que chaque T.E. ne doit pas être relié à l'un d'eux; par ailleurs, si le concepteur désire introduire et gérer une redondance, il peut associer un même T.E. à plusieurs fichiers.

### 4.1.3.4. L'IKO

Au terme de la description du modèle conceptuel (4.1.2.), nous avons notamment remarqué le manque de représentation de certaines notions liées à l'identifiant et de l'accès aux données. En utilisant le formalisme MAG ( (17) , annexes 5.1 et 5.2 ), définisons les éléments suivants.

#### Les items identifiants

"...Un identifiant est un item (ou une liste d'items) d'un type d'article tel qu'il n'existe pas, dans le référentiel précisé, plus d'un article qui soit associé à une même valeur de cet item (ou des items de cette liste). Un identifiant est caractérisé par le référentiel dans lequel il porte ses effets...." (Hainaut)

En particulier, nous considérons les référentiels suivants :

"- tous les articles de la base de données appartenant à un type déterminé
- tous les articles cibles d'un chemin appartenant à un type d'articles déterminé." (Hainaut)

Pour limiter notre représentation tout en la maintenant cohérente, nous pouvons associer un identifiant :

- au type d'article (unique) sur lequel il porte (ici, au T.E.)
- à (aux) l'item(s) du type d'article considéré s'il en existe qui font partie de l'identifiant (ici, au(x) ATTR)
- à (aux) type(s) de chemin(s) reliant le type d'article considéré à d'autres s'il en existent dont les items font partie de l'identifiant (ici, au(x) TAS)

Nous devons ajouter comme contrainte qu'un identifiant doit etre relié au moins à un ATTR et/ou au moins à un TAS.

## Les clés d'accès

"...Une clé d'accès est un item (ou une liste d'items) d'un type d'article tel qu'il existe un mécanisme qui permette d'accéder successivement aux articles auxquels est associée une valeur déterminée de cette clé, et à eux seulement. Une clé d'accès est caractérisée par le référentiel dans lequel elle porte ses effets..."(Hainaut)

Le raisonnement est similaire à celui qui est fait pour l'identifiant. Nous pouvons donc établir un parallèle entre ces deux notions bien distinctes.

## L'ordre des articles cibles d'un chemin d'accès

"...L'accès aux articles cibles offert par le mécanisme du chemin d'accès étant de nature essentiellement séquentielle (accès aux cibles successives), il convient dans certains cas de pouvoir spécifier la règle, si elle existe, qui définit l'ordre des articles cibles dans la séquence d'un chemin..." (Hainaut)

Nous considérons seulement le cas ou les types de chemins possèdent un seul type de cible. On distingue quatre types d'ordres :

- "Si l'ordre est indifférent, l'endroit d'insertion d'une cible dans le chemin est laissé à l'initiative du système de gestion.

- L'ordre choisi peut être lié au moment de l'insertion:
  - chronologique : le premier article inséré est le premier du chemin (queue, file d'attente,liste FIFO)
  - antichronologique : le dernier article inséré est le premier du chemin (pile,liste LIFO)

- L'ordre peut être trié selon les valeurs d'une clé de tri; cette clé est constituée d'un ou plusieurs items liés au type des articles cibles. Les articles sont accessibles dans l'ordre des valeurs croissantes (décroissantes) du premier item de la clé; en cas de doubles de cette valeur, les articles concernés sont accessibles dans l'ordre des valeurs croissantes (décroissantes) du deuxième item de la clé; et ainsi de suite.

Si la clé de tri n'est pas un identifiant dans le chemin d'accès, les articles de même valeur de cette clé de tri seront rangés dans l'ordre chronologique ou antichronologique..." (Hainaut). Et nous pouvons ajouter : ou dans un ordre indifférent c'est-à-dire laissé à l'initiative du système de gestion.

- "Lorsque ces ordres prédéfinis ne conviennent pas, il est possible de laisser au programme d'application le soin de déterminer le point d'insertion dans le chemin..." (Hainaut)

De tout ceci découle que nous associerons cet ordre :

- au type d'article unique cible d'un type de chemin d'accès (ici, au T.E.)

- au type de chemin éventuel dont les articles sont la cible (ici, au TAS)

- aux items (éventuels) constituant la clé de tri (ici, au(x) ATTR); nous préciserons en outre, dans chacune de ces dernières associations, le sens du tri (croissant ou décroissant) et le numéro d'ordre de ce composant de la clé de tri (numéro de séquence).

Nous donnerons à cet ordre les propriétés suivantes :

- type de l'ordre:
  - indifférent = R (comme "random")
  - FIFO = F
  - LIFO = L
  - trié = S (comme "sorted")
  - programmé = P
- ordre des doubles (si la clé de tri n'est pas identifiante)
  - indifférent = R
  - FIFO = F
  - LIFO = L

L'ordre d'accès associé à une clé d'accès
---

"...Cet accès, lorsqu'il s'agit d'une clé non identifiante, peut conduire à plus d'un article; il convient par conséquent de pouvoir spécifier la régle, si elle existe, qui définit l'ordre dans lequel les articles sont parcourus et livrés. Cette propriété est strictement analogue à l'ordre des articles cibles d'un chemin d'accès..." (Hainaut)

Nous devons cependant préciser que, dans ce cas, l'ordre (et la clé) est associé :

- au type d'article (unique) sur lequel sont définis l'ordre et la clé d'accès (ici, au TE)
- à (aux) item(s) (éventuels) constituant la clé de tri (ici,aux ATTR) en faisant les mêmes précisions que précédemment
- aux autres éléments propres à la définition de la clé d'accès.

Regroupement de toutes ces notions
---

Pour faire l'économie d'un grand nombre de T.E. et de T.A., regroupons toutes ces notions sous le terme IKO "(identifier, key, order)". Nous préciserons pour celui-ci :

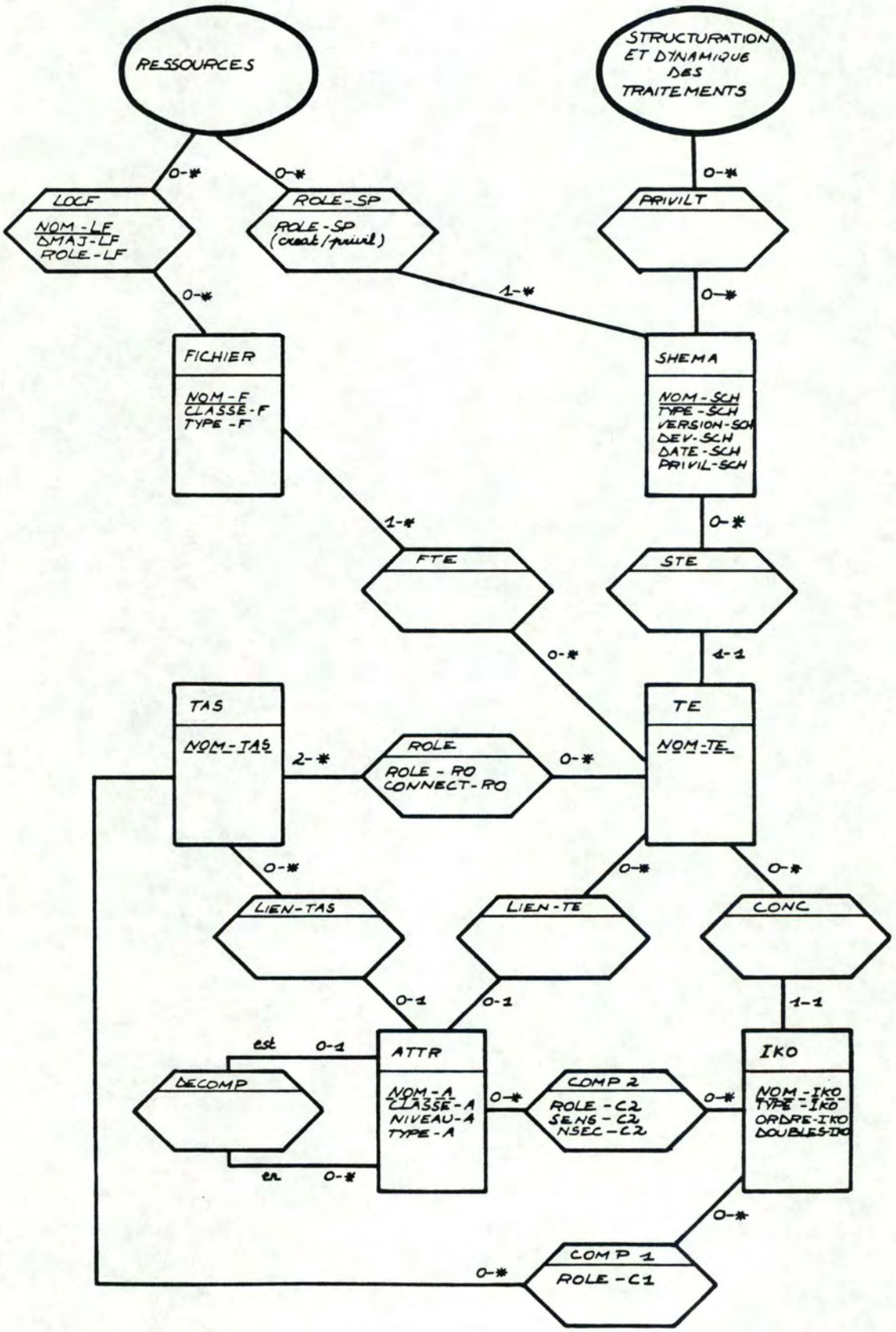
- le type : I (identifiant), K (clé d'accès), O (ordre des articles cibles d'un chemin d'accès), KO (ordre d'accès associé à une clé d'accès) ou une combinaison de ceux-ci (IO, IK, IKO)
- l'ordre : (si l'IKO contient O) : R, F, L, S, P
- les doubles : (si l'IKO ne contient pas I) : R, F, L
- le nom : identifiant chaque IKO associé à un même T.E.

Chaque IKO est défini sur un et un seul T.E.; ajoutons le T.A. CONC (concerne). Les composants d'un IKO sont toujours formés d'un ou plusieurs TAS et/ou d'un ou plusieurs ATTR. Ajoutons respectivement les T.A. COMP1 et COMP2 en précisant le rôle joué par chaque composant. Si l'ordre de l'IKO est trié (S), celui-ci est relié via COMP2 à un ou plusieurs attributs formant une clé de tri. Dans ce cas, nous précisons en outre, dans l'association, le numéro de séquence (le numéro de composant dans la clé de tri) et le sens du tri (croissant ou décroissant).

## 4.1. 4 Sous-modèle de structuration des données définitif

Introduisons la notion d'IKO dans les schémas intermédiaires précédents (4.1.2 et 4.1.1.3), nous obtenons le sous-modèle définitif.

Schéma conceptuel



## 4.2. STRUCTURATION ET DYNAMIQUE DES TRAITEMENTS

### 4.2.1.Approche générale

Pour donner une signification à la notion de traitement que nous introduisons ici, nous allons la placer dans le contexte général de développement d'un projet informatique. Le développement d'un projet de quelque importance ne se ramène pas au seul codage dans un langage plus ou moins évolué. Bien plus, les études actuelles définissent le cycle de vie d'un projet,qui se divise en grandes étapes dont une des dernières est le codage.

Nous allons brièvement considérer le développement et l'organisation des traitements pour chacune de celles-ci. Nous nous limitons aux éléments utiles à l'organisation considérée: l'UCM de Namur.

### 4.2.2.L'analyse conceptuelle

L'analyse conceptuelle d'un projet, la première étape du cycle de vie, posséde une vue très large du S.I.: elle considère les traitements manuels et automatisables. A priori, nos ambitions sont plus limitées mais nous en retiendrons cependant certaines notions.

#### 4.2.2.1.Structuration des traitements

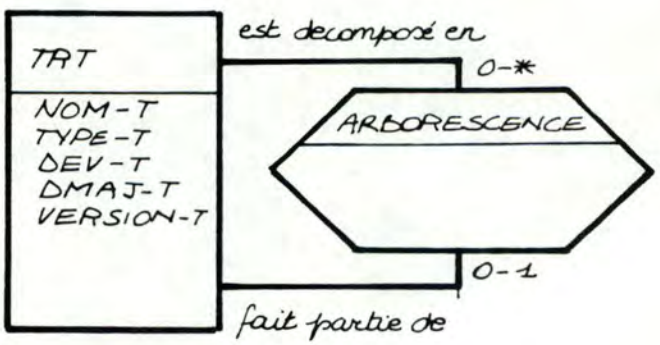
Un modèle de structuration des traitements, développé dans (11) a pour objectif :
"... de fournir aux concepteurs et analystes des critères leur permettant de décomposer un projet en traitements de plus en plus élémentaires. Plus précisément, ce modèle devrait procurer des critères d'identification de ces traitements en fonction d'une nomenclature standardisée, chaque niveau de celle-ci assumant un rôle particulier dans la conception d'un S.I..." (Bodart)

Parmi les propriétés générales du modèle, nous retiendrons que :
"...Tout traitement est décomposé sous forme arborescente : un traitement de niveau intermédiaire i($i > 1$) provient de la décomposition d'un seul traitement de niveau i-1 et se décompose en n($n \geqslant 1$) traitements de niveau i+1. (...)
Cette décomposition met en évidence le fait que chaque traitement (sauf le traitement initial) fait partie d'un traitement de niveau supérieur...." (Bodart)

Les différents niveaux sont définis par les concepts de projet, application, phase et fonction. Nous ne détaillerons ici ni leur définition ni leurs critères d'identification propres. Nous considèrons seulement cette première approche de la notion de traitement. Dans notre modèle, nous les identifierons par un nom, nous leur associerons un type (sans imposer de nomenclature particulière), une date de dernière mise à jour, un numéro de version éventuel et son stade de développement (en développement ou terminé). La décomposition arborescente sera traduite par un T.A. recursif avec ses connectivités.

Remarquons qu'il a été admis au départ que nous pouvions associer à toute occurrence une ou plusieurs lignes de texte (cfr chap.3). Nous laissons à l'utilisateur le soin de se servir de cette possibilité pour définir ou donner les spécifications fonctionnelles d'un traitement (ou toute autre description). De cette façon, nous faisons l'économie d'un ou plusieurs attributs.

Schéma conceptuel



#### 4.2.2.2. Dynamique des traitements.

Toujours au niveau conceptuel, une méthode d'analyse prévoit de représenter le comportement du S.I. suivant un modèle de la dynamique des traitements. (11)

"...L'objectif de ce modèle est de fournir à l'analyste des concepts et des mécanismes lui permettant de représenter les conditions de déclenchement, d'exécution et d'enchainement des traitements en vue de caractériser les éléments du S.I. qui causent la production des messages-résultats et les changements d'état de la mémoire du S.I..." (Bodart).

Dans la mesure où ce modèle a pour principale utilité de produire une maquette du S.I., nous ne considérerons pas ces notions pour nous limiter aux architectures logiques et physiques (cfr 4.2.3 et 4.2.4) qui prennent en compte cette notion de dynamique.

Néanmoins, le concept très général de traitement que nous développons peut être enrichi des notions propres à la dynamique au niveau conceptuel (processus, événement, condition,...).
L'utilisateur désireux d'étendre son modèle à ces différents concepts a donc toute liberté de le faire grace au principe d'extensibilité que nous nous sommes donné au départ.

### 4.2.3. Conception d'une architecture logique.

Au départ de l'analyse conceptuelle qu'un utilisateur a peut-etre réalisé à l'aide du dictionnaire, celui-ci décrit ensuite la conception d'une architecture logique, c'est-à-dire la structuration du système en un ensemble de composants. (Ceux-ci sont issus des traitements automabisables ou semi-automabisables définis à l'étape précédente). Dans notre dictionnaire, nous les appelerons également des traitements, (ce qui en élargit une première fois la définition).

La littérature développe de nombreuses méthodes de distinction et de structuration des traitements à ce niveau. Nous allons aborder les deux grandes démarches possibles pour élargir les notions de notre modèle.

Hiérarchiser un système:
---

"...Structurer un système de manière hiérarchique, c'est l'organiser suivant des niveaux distincts et ordonnés. Dans toute structure hiérarchique, il y a une relation d'ordre.(...)
Une structure est hiérarchisée si et seulement si il existe une relation R entre ses composants permettant de définir des niveaux tels que:

$niv_0 = \{ A \mid \nexists B \text{ tq } R(A,B) \}$

(= ensemble des composants A pour lesquels il n'existe aucun composant B tel que R (A,B))

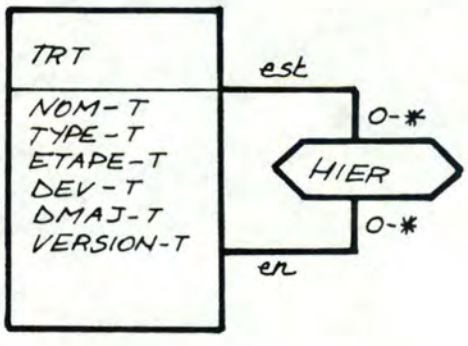
$niv_i = \{ A \mid 1) \exists B \text{ de } niv_{i-1} \text{ tq } R(A,B)$
$2)$ si on a R(A,C) pour certains composants C, alors C est de niveau k avec $0 \leqslant k \leqslant i-1 \}$

On peut définir autant de systèmes hiérarchiques que l'on peut définir de relations R..."
(Notes prises au cours de méthodologie de développement de logiciels donné par A. Van Lamsweerde)).

Pour représenter une telle structure, nous généralisons la notion d'arborescence à celle de hiérarchie (sans pour autant interdire à l'utilisateur de restreindre son utilisation). La distinction des différents niveaux est laissée aux soins de l'utilisateur qui définira sa nomenclature.

Pour préciser l'étape de développement du projet auquel le traitement appartient, nous ajoutons un attribut qui la décrit de manière informelle (analyse conceptuelle, architecture logique, ...)

Schéma conceptuel.



Ceci modifie les connectivités de notre schéma et supprime certaines contraintes liées au T.A. récursif. Nous appelerons celui-ci hiérarchie pour plus de clarté.

Remarquons que certaines méthodes de structuration hiérarchique se distinguent uniquement par la sémantique qu'elles donnent à la relation R. Celle-ci peut être par exemple: "utilise", "alloue des ressources à", "est accessible à partir de", "est partie de" (relation définie dans l'approche précédente). Ceci ne modifie en rien notre schéma; toute description supplémentaire peut être donnée via le T.E. TEXTE.

Structuration modulaire d'un système.
---

Alors que la hiérarchie définit l'organisation des niveaux dans un système, la modularisation spécifie en plus la découpe des composants d'un meme niveau. Sans entrer dans le détail, rappelons une définition largement reconnue d'une structure modulaire:

"...Une structure est modulaire si les composants qui la forment et les relations qui existent entre ses composants sont tels que:
- les attributs de chaque composant peuvent être définis de manière simple et précise,
- chaque composant offre:
  - une forte capacité de cacher de l'information
  - une forte cohésion interne
  - un faible degré de couplage..."
  (Van Lamsweerde)

Cette définition nous permet d'élargir à nouveau la notion de traitement à celle de module. Outre les types de relations définis précédemment, les modules d'un même niveau peuvent être également associés. Dans ce cas, on parlera généralement d'exportation ou d'importation entre modules. En restreignant à nouveau nos contraintes sur le T.A. HIER (hiérarchie), nous pouvons traduire ce type de relation. Bien que le terme de "hiérarchie" ne soit pas adéquat,dans ce dernier cas, (nous pourrions parler d'"organisation"), nous le conservons pour la signification générale que nous lui avons définie jusqu'à présent (et qu'elle conservera généralement dans la suite.) Nous n'apportons donc aucune modification au schéma précédent.

### 4.2.4. Conception d'une architecture physique.

Cette dernière grande étape précédant le codage proprement dit définit l'organisation des modules physiques dont les choix de découpe sont influencés par les contraintes du site d'implémentation (langages de programmation disponibles, OS, SGBD,...).

Pour ce stade du développement, l'UCM a défini sa propre organisation et suit une terminologie précise. Ceci est notamment expliqué dans le dossier informatique de la CAS (annexe 3.1., chapitre 3). Rappelons les différents types de modules physiques: application, module, sous-module, opération, programme et écran.

L'organisation est hiérarchique avec utilisation multiple des niveaux inférieurs. Cette structure pouvant etre modélisée dans notre schéma sans modification, nous étendons simplement la notion de traitement aux modules physiques. Les utilisateurs pourront pour les traitements de cette étape choisir leurs propres contraintes, par exemple en n'autorisant pas d'autres types que ceux qu'ils ont défini.

### 4.2.5. Codage.

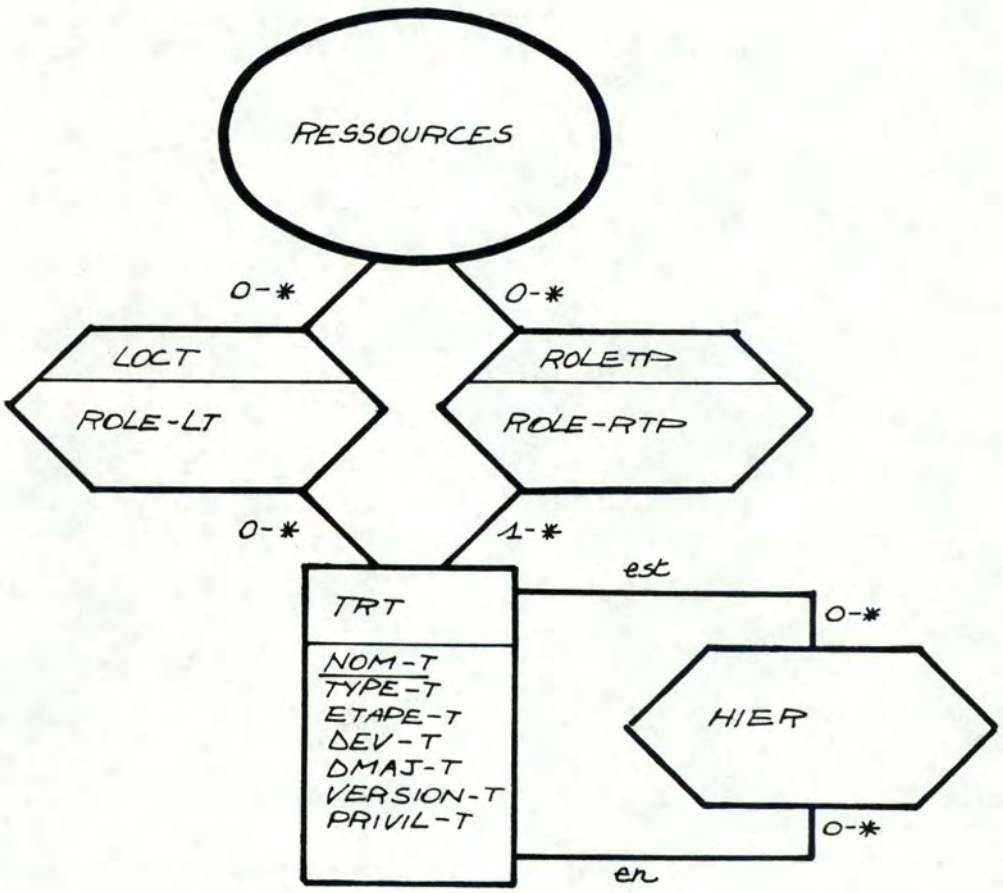
Nous ne détaillerons pas cette étape bien connue des informaticiens, ni les suivantes (tests, maintenance) qui sortent du cadre de notre modèle. Nous préciserons simplement qu'une unité de code source ou objet peut être associée à un traitement d'une architecture physique. Pour décrire cette notion, nous ajoutons le T.A.: LOCT (localisation d'un traitement) reliant tout traitement à un élément du sous-modèle des RESSOURCES. Il sera défini par un attribut précisant le rôle joué par chaque localisation (version ou source, test, exécutable).

### 4.2.6. Notions supplémentaires.

Par analogie avec le sous-modèle des données, nous ajouterons pour être complet un T.A.: ROLETP, reliant tout traitement à son créateur (obligatoire et unique) et, éventuellement, aux utilisateurs qui ont un privilège d'accès si celui-ci n'est pas publique. Un attribut de l'association précise quelle est la nature du lien. Cette notion de privilège est en outre définie par un attribut supplémentaire du traitement (accès publique ou non).

Nous obtenons le sous-modèle définitif de la structuration des traitements en introduisant ces notions au dernier schéma.

Schéma conceptuel



## 4.3. STATIQUE DES TRAITEMENTS.

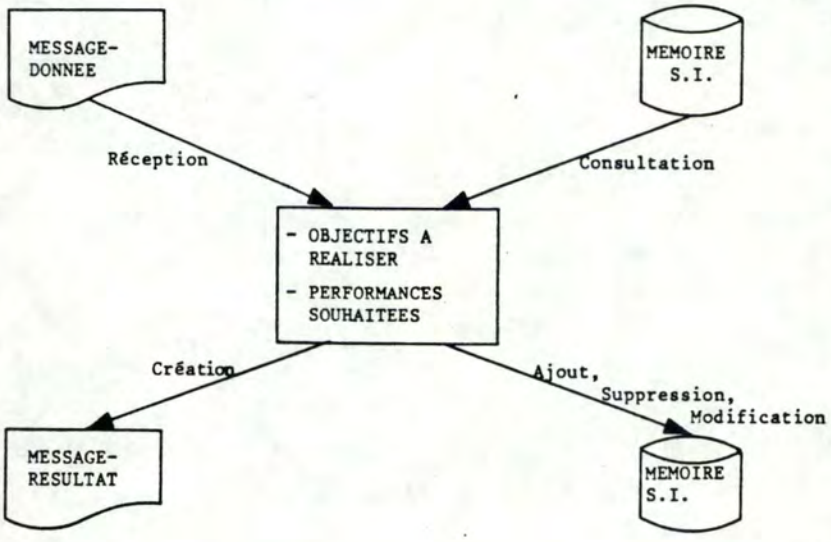
La description de la statique des traitements rassemble des notions qui constituent une charnière dans notre modèle.
Nous possédons assez d'informations pour les introduire à ce stade et ainsi réaliser un premier lien entre les sous-modèles développés.

Repérons d'abord les éléments présentés ici et introduisons-les ensuite dans notre schéma.

### 4.3.1. Le modèle de la boite noire.

Présenté comme composant du modèle de la statique des traitements dans une méthode d'analyse conceptuelle (11), il est développé comme ceci.
"...Ce modèle permet de spécifier pour un traitement d'un niveau donné:

- les objectifs à réaliser,
- les performances souhaitées,
- les informations en entrée et en sortie,
- les actions primitives qui caractérisent les relations entre ces informations et le traitement..." (Bodart)



Nous ne considérerons pas le concept de message qui nous parait plus spécifiquement lié à l'analyse conceptuelle. Par contre, ce modèle nous est particulièrement utile pour définir la liaison entre une structure de données (mémoire S.I.) et un traitement qui peut faire:

- des consultations (entrées)
- des ajoutes, des suppressions et des modifications (sortie)

Cette description peut en outre être précieuse à tous les stades de développement d'un projet, y compris l'architecture physique.

La représentation d'une information de ce type est par définition un lien entre les sous-modèles STRUCTURATION ET DYNAMIQUE DES TRAITEMENTS d'une part et STRUCTURATION DES DONNEES d'autre part. Ce lien, que nous appellons I-O ("input-output"), est caractérisé par le rôle joué par chaque structure de données au sein d'un traitement.

Celui-ci sera défini par l'utilisateur suivant les termes qu'il désirera (exemple: consultation, création, destruction, modification, insertion, suppression).

### 4.3.2. Lien entre les traitements et les données.

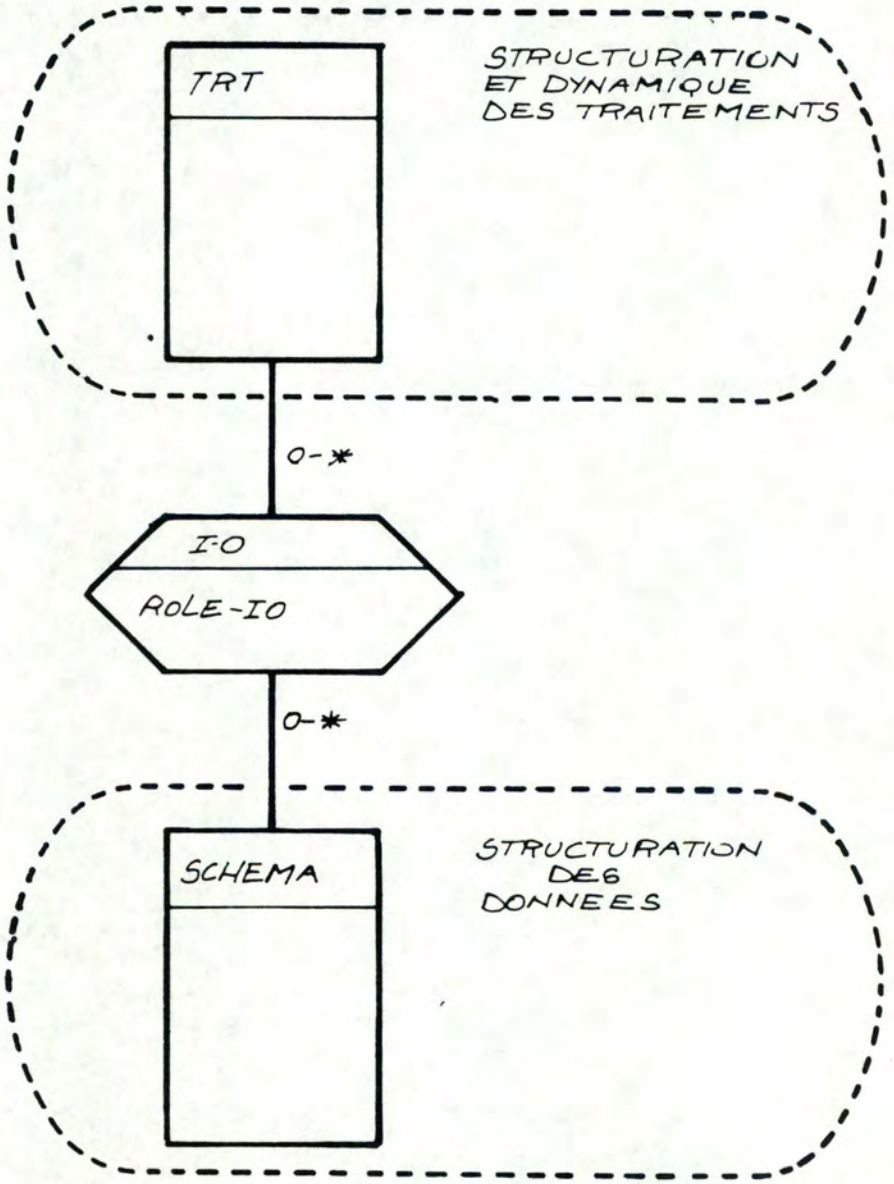
Représentons le lien I-O sous forme d'un T.A. D'une part, il est évident qu'il sera associé au traitement au vu de la sémantique très large que nous avons donnée à ce T.E. D'autre part, pour la structure des données, il peut être relié soit au T.E. SCHEMA, soit au T.E. FICHIER, soit à chacun de ceux-ci. Nous nous limiterons à un T.A. binaire entre TRT et SCHEMA au vu du role central que joue ce dernier dans les données.Rappelons que si un niveau de précision plus fin est requis, il est possible d'ajouter une occurence à SCHEMA qui décrira un seul FICHIER ou même une seule partie d'un FICHIER ou d'un T.E.

Pour terminer notre unification des deux sous-modèles, remarquons qu'un autre lien était prévu entre SCHEMA etTRT, décrivant les privilèges d'accès par un traitement sur une structure de données (notion introduite après l'examen des tables système du SGBD SQL).

Comme nous avons par ailleurs représenté les privilèges que le créateur d'un traitement peut accorder à d'autres utilisateurs (cfr : 4.2.5.), nous nous limiterons à cette dernière représentation au risque de traduire moins clairement des informations dans certains cas (que nous imaginons très peu fréquents à l'UCM).

Ces dernières remarques et les notions liées à la statique des traitements nous permettent de relier définitivement les deux sous-modèles

Shéma conceptuel



## 4.4. RESSOURCES.

### 4.4.1. Notion générale.

La notion de ressources est généralement très développée lors de l'analyse conceptuelle d'un projet. Un modèle des ressources tente
"...d'estimer si une solution conceptuelle est réalisable compte tenu de la disponibilité des ressources de l'organisation, telles que, par exemple, le personnel, les équipements, les moyens financiers, les horaires de travail ou la composition des équipes et des postes de travail..." (Bodart)

Inversément, les DD que nous avous consultés sont souvent très pauvres pour décrire ce type d'information (notion de "USER" représentant un utilisateur ou un groupe d'utilisateurs dans l'IDD de CULLINET); certains ne la considérant pas du tout.

Par ailleurs, notre modèle est spécifique à une organisation (UCM) qui possède des besoins en documentation qui sont très peu définis en ce domaine. Nous limiterons donc notre schéma aux grandes notions dont le besoin a été ressenti aux étapes précédentes (T.A. vers les ressources) en les particularisant suivant l'organisation de l'UCM.

### 4.4.2. Deux types de ressources.

D'une part, le créateur d'un schéma ou d'un traitement et les privilèges que celui-ci accorde à d'autres utilisateurs pour leurs accès englobent le concept de ressources en personnel. D'autre part, une localisation physique d'un fichier ou d'un code d'un traitement fait partie des ressources matérielles d'une organisation.

### 4.4.3. Ressources en personnel.

Considérons très classiquement qu'une personne est un membre du personnel de l'UCM dépendant d'un des quatre services (cfr. chapitre 2) ou du service général (le directeur général, sa secrétaire, le directeur du service informatique) et à propos de laquelle on désire décrire son activité dans l'organisation.

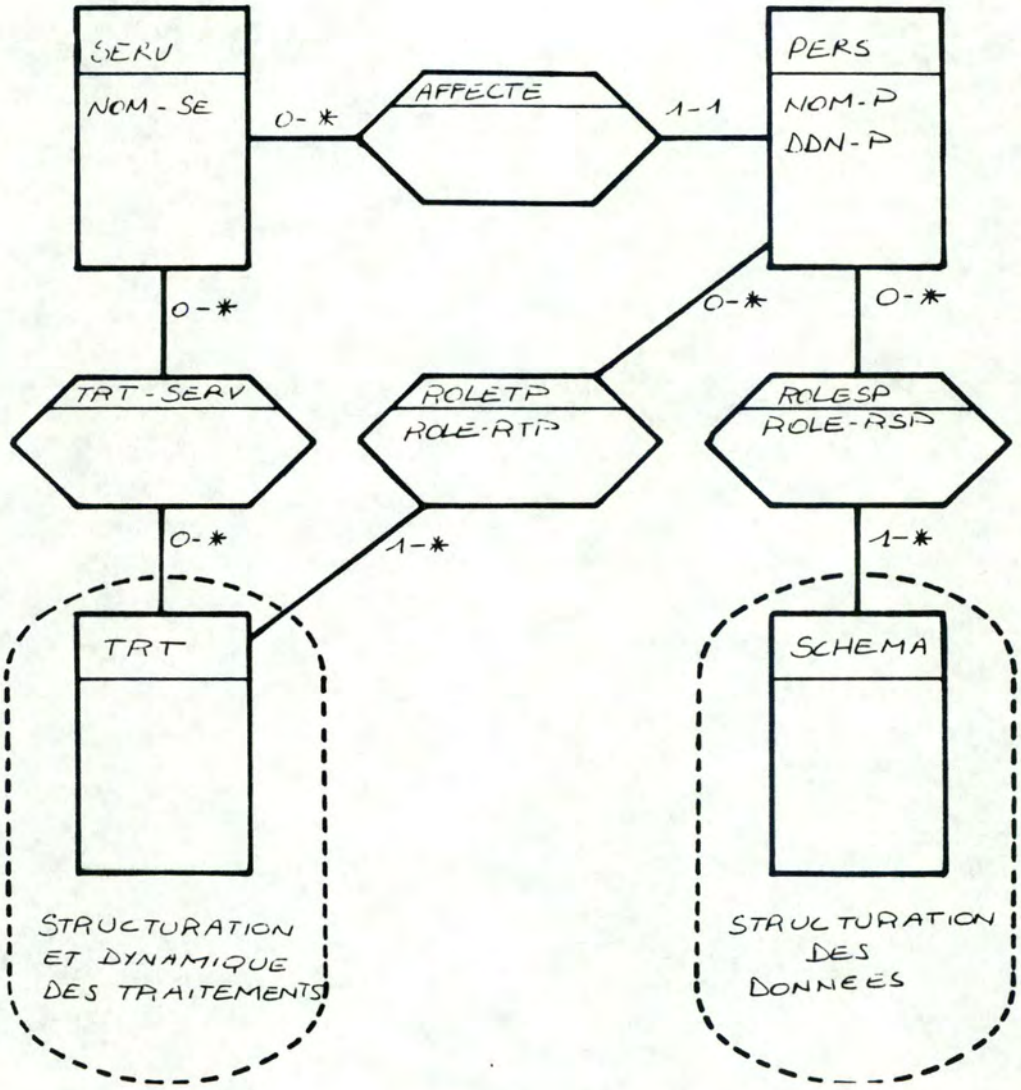
C'est ce T.E. qui sera le second membre du T.A. ROLETP (création ou privilège sur un ou plusieurs traitements) et du T.A. ROLESP (idem sur un ou plusieurs schémas).

Nous le caractérisons en outre par un nom identifiant et une date de naissance.

Chaque membre du personnel est financièrement et hiérarchiquement affecté à un et un seul des services de l'UCM (un des quatre services ou le service général): nous caractériserons ces derniers par leur nom.

Nous savons en outre qu'une personne peut exceptionnellement être lié à des traitements utiles à d'autres services que celui dont elle dépend. Nous traduirons ce dernier point par un T.A. reliant service à traitement sans aucune contrainte d'obligation ou de connectivité (pour ne pas introduire de redondance obligatoire avec hiérarchie et pour permettre à certains traitements d'etre utilisés par plusieurs services, bien que ce cas ne se soit pas encore présenté à l'UCM.)
Nous obtenons le schéma de la description des ressources en personnel.

Schéma conceptuel (voir page suivante)




### 4.4.4. Ressources en matériel.

Appelons mémoire auxiliaire (MEM-AUX), toute unité physique de stockage de l'information.
Celle-ci servira à repérer sur quel disque ou bande magnétique se situe cet exemplaire d'un fichier ou d'un programme (plus généralement d'un traitement).

Nous donnerons pour chacune de celle-ci: un nom identifiant, un type (disque, bande, disquette), et une classe (pour une description plus précise comme par exemple: fixe ou amovible).

Ceci laisse toute possibilité à l'utilisateur de définir une description plus complète par extensibilité ou association de lignes de texte (par exemple pour l'organisation du rangement des bandes magnétiques servant pour l'archivage).

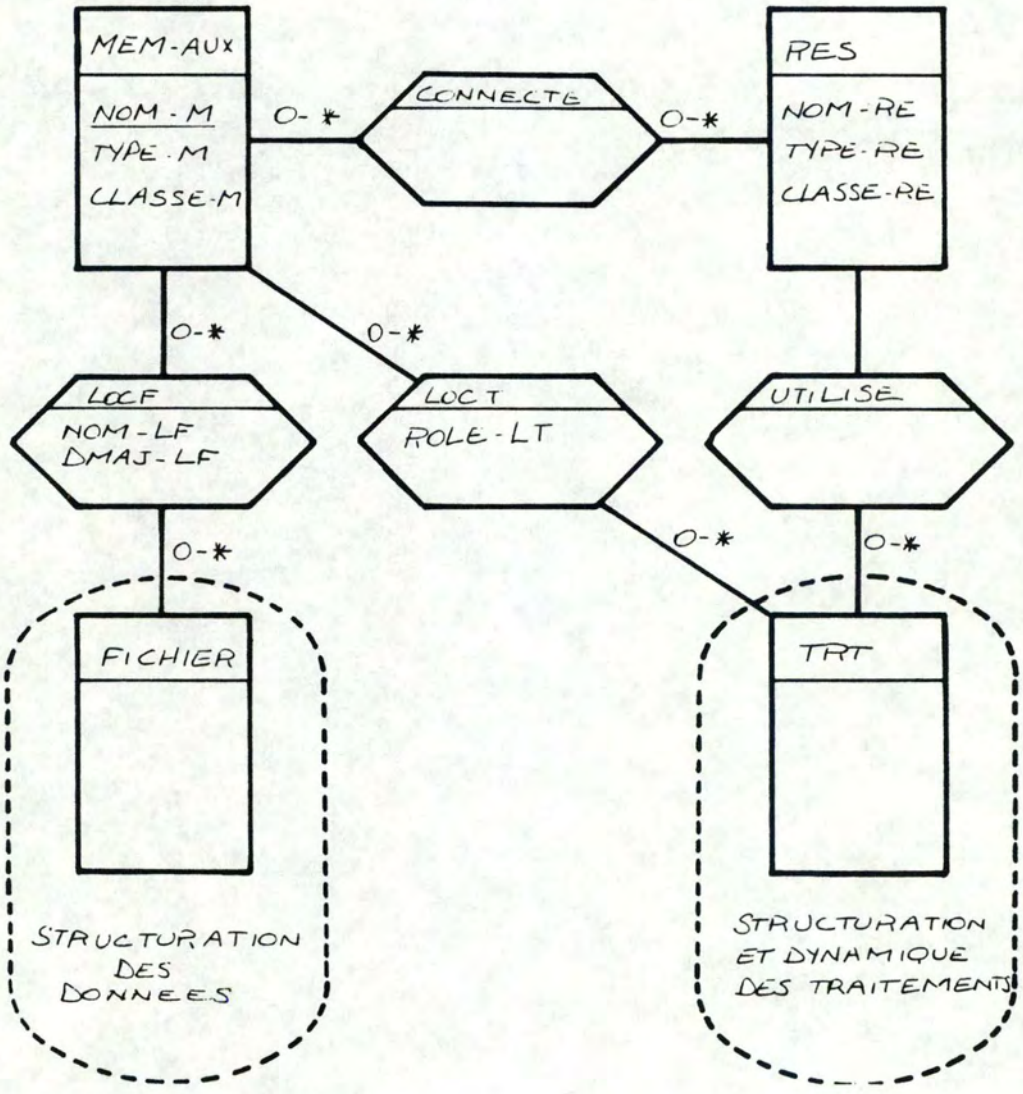
Pour être complet, nous avons regroupé dans un dernier T.E. toute autre ressource matérielle ("hard") ou logicielle ("soft") du SI. Celle-ci est définie de la meme façon que la mémoire auxiliaire (nom identifiant, type, classe). Plutot qu'une unité de description, ce dernier T.E. résume un dernier ensemble de notions bien distinctes.

Il joue un rôle informatif sur la façon de le relier aux autres éléments du schéma, celui-ci comprenant au minimum: une appartenence de la mémoire auxiliaire aux ressources et l'utilisation de celles-ci par un traitement quelconque.

L'utilisateur désireux de les particulariser dispose de l'extensibilité pour étendre ou modifier son schéma. Ainsi par exemple, une des premières versions considérait la notion de machine virtuelle (chapitre 2) associé à une ou plusieurs personnes (suivant l'organisation à l'UCM) et par spécialisation aux ressources en matériel.

Notre dernier sous-schéma est ainsi défini. Ceci termine la description du schéma conceptuel du dictionnaire de données.

Schéma conceptuel. (voir page suivante)


MEM-AUX
NOM-M
TYPE-M
CLASSE-M
CONNECTE
RES
NOM-RE
TYPE-RE
CLASSE-RE
0-*
0-*
0-*
0-*
LOCF
NOM-LF
DMAJ-LF
LOCT
ROLE-LT
UTILISE
0-*
0-*
0-*
FICHIER
TRT
STRUCTURATION DES DONNEES
STRUCTURATION ET DYNAMIQUE DES TRAITEMENTS

## 4.5. RESUME DU SCHEMA CONCEPTUEL COMPLET

### MEM-AUX (mémoire auxiliaire)

Une occurrence définit une unité physique de stockage, comme un disque ou une bande magnétique,...

NOM-M
nom de l'unité de stockage

TYPE-M
type de stockage (bande, disque, disquette,...)

CLASSE-M
une des deux valeurs:
- FX: si l'unité est en permanence sur le lecteur
- AM: si elle est amovible

identifiant: NOM-M

### RES (ressources)

Une occurrence décrit une ressource matérielle ("hard") ou logiciel ("soft") du système d'information.

NOM-RE
nom de la ressource

TYPE-RE
un des types suivant: (exemple: "hard", "soft")

CLASSE-RE
description informelle ("hard": ordinateur, lecteur disque, terminal, imprimante, ..., "soft": éditeur, SGBD, compilateur, OS, ...)

identifiant: NOM-RE

### SERV (service)

Une occurrence correspond à un service de l'UCM c.à.d. CAS, SS, CCAF, PME et GENERAL (direction générale)

NOM-SE
nom du service

identifiant: NOM-SE

PERS (personne)
---

Une occurrence correspond à un membre du personnel de l'UCM dépendant d'un des quatre services ou du service général et à propos duquel on désire décrire l'activité dans l'organisation.

NOM-P
nom de la personne

DDN-P
date de naissance de la personne

identifiant: NOM-P

TRT (traitement)
---

Une occurrence représente une action simple ou composée sur le SI à tout niveau de conception.

NOM-T
nom du traitement

TYPE-T
description informelle (exemple: application, module, sous-module,...)

ETAPE-T
description informelle (exemple: conception, spécification, exécutable,...)

DMAJ-T
date de dernière mise à jour

DEV-T
en développement (D) ou terminé (T)

VERSION-T
numéro de version: 1 ou > 1 (si plusieurs versions sont conservées)

PRIVIL-T
- N: si accès à tous les utilisateurs
- O: si un ou plusieurs privilèges sont accordés

identifiant: NOM-T

### LOCT (localisation traitement)

Une occurrence définit l'emplacement physique d'un traitement

ROLE-LT
description informelle (exemple: source, objet, spécification,...)

### ROLETP

Une occurrence définit le role joué par une personne vis-à-vis d'un traitement

ROLE-RTP
- 'CREAT': créateur
- type de privilège: 'LECTURE', 'ECRITURE', 'MODIF'

### SCHEMA

Un schéma regroupe un ensemble de définitions d'une structure de données à une étape quelconque de sa conception.

NOM-SCH
nom du schéma

TYPE-SCH
description informelle (exemple: schéma conceptuel, schéma des accès possibles, schéma interne,...)

VERSION-SCH
numéro de version: 1 ou $>$ 1 (si plusieures versions sont conservées)

DEV-SCH
en développement (D) ou terminé (T)

DATE-SCH

date de dernière mise à jour

PRIVIL-SCH
- N: si accès à tous les utilisateurs
- O: si un ou plusieurs privilèges sont accordés

identifiant: NOM-SCH

### ROLESP

Une occurrence définit le role joué par une personne vis-à-vis d'un schéma

ROLE-RSP
- 'CREAT': créateur
- type de privilège: 'LECTURE', 'ECRITURE', 'MODIF'

### FICHIER

Une occurrence définit le regroupement logique de certains éléments d'un schéma en fonction des possibilités d'un SGD particulier.

NOM-F
nom du fichier

CLASSE-F
suivant la classe d'utilisation du fichier: 'PERM' (permanent), 'TEMP' (temporaire)

TYPE-F
description informelle (exemple: séquentiel indexé, table SQL,...)

identifiant: NOM-F

### LOCF (localisation fichier)

Une occurrence définit un enregistrement physique particulier d'un fichier

NOM-LF
nom du fichier physique

DMAJ-LF
date de dernière mise à jour

ROLE-LF
description informelle (exemple: maitre, copie, version)

### T.E. (type d'entité)

Toute occurrence correspond à une collection d'entités ou d'articles (cfr. définitions annexe 5.1.4.2.)

NOM-TE
nom du type d'entité

identifiant: NOM-TE, NOM-SCH

### TAS (type d'association)

Toute occurrence correspond à une collection d'associations ou de chemins d'accès (cfr. définitions annexe 5.1.4.2.)

NOM-TAS
nom du type d'association

identifiant: NOM-TAS, NOM-SCH

### ROLE

Une occurrence définit le rôle particulier que joue un T.E. dans un TAS.

ROLE-RO
description informelle (exemple: origine, destination,...)

CONNECT-RO
connectivité du T.E. vis-à-vis du TAS

### ATTR (attribut)

Toute occurrence correspond à un ensemble possible de valeurs d'attribut ou de valeurs d'item (cfr. définitions annexe 5.1.4.2.).

NOM-A
nom de l'attribut

CLASSE-A
reprend les propriétés générales d'un attribut (exemple: OES, FDR, ...)

NIVEAU-A
numéro de niveau dans l'arborescence

TYPE-A
type de valeur de l'attribut (exemple: caractère, entier,...) ainsi que son format (exemple: nombre de caractères, emplacement de l'exposant,...)

identifiant: NOM-A, id.(T.E. ou TAS)

IKO ("identifier", "key", "order")
---

Une occurrence représente un ou plusieurs des composants: identifiant, clé d'accès, ordre des articles cibles d'un chemin d'accès, ordre d'accès associé à une clé d'accès.

NOM-IKO
nom de l'IKO

TYPE-IKO
un des types suivants: I, K, O, KO, IO, IK, IKO

ORDRE-IKO
un des ordres (si l'IKO contient O): R, F, L, S, P

DOUBLES-IKO
ordre des doubles(si l'IKO ne contient pas I): R, F, L

identifiant: NOM-IKO, id.(T.E.)

COMP1 (composant)
---

ROLE-C1
définit le role joué par un TAS dans un IKO

## COMP2 (composant)

ROLE-C2
définit le role joué par un ATTR dans un IKO

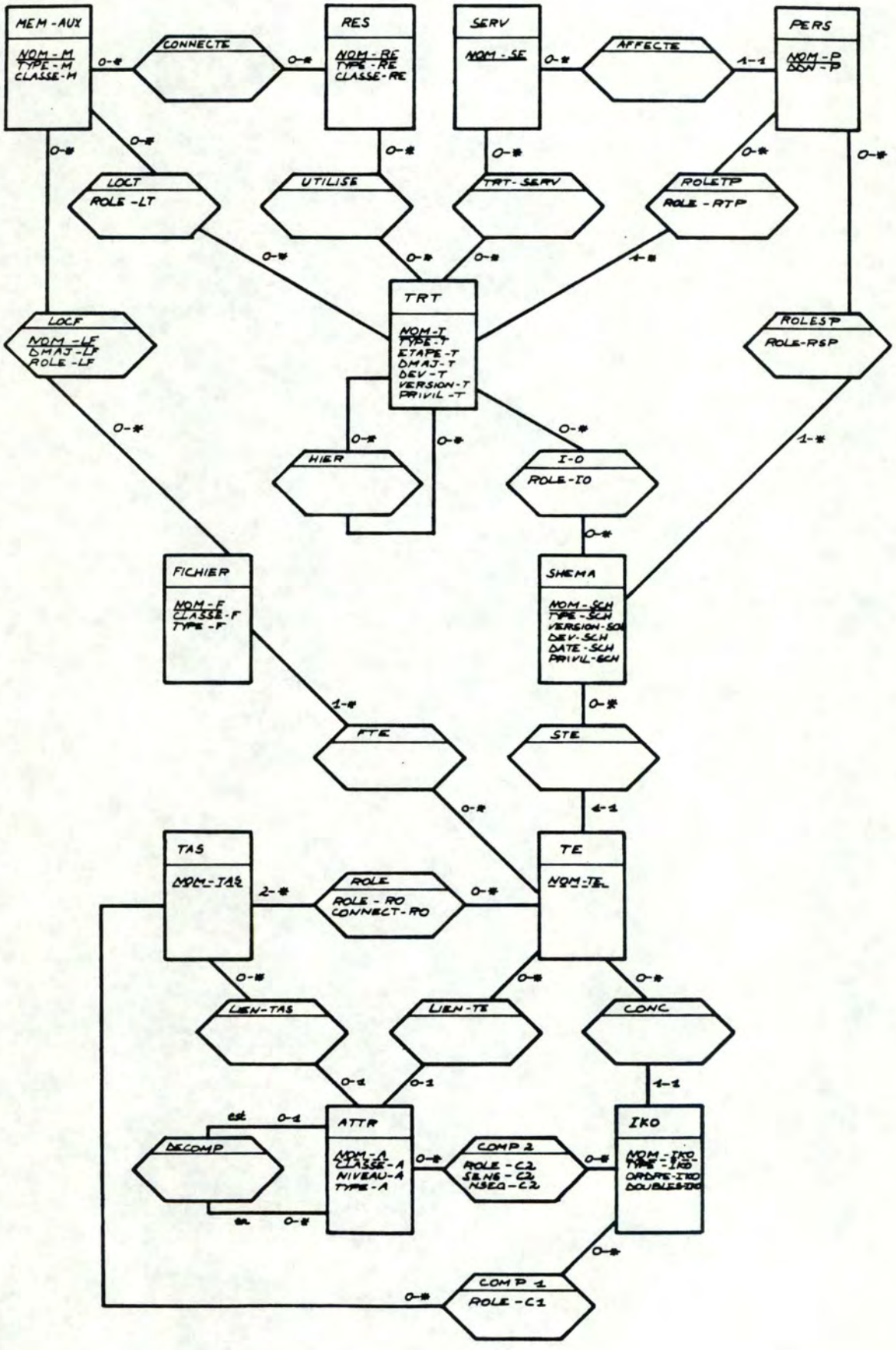
SENS-C2
(éventuel) définit l'ordre d'accès d'un composant d'une clé de tri: C (croissant) ou D (décroissant)

NSEQ-C2
(éventuel) numéro de séquence de cet attribut dans la composition de la clé de tri

## IO ("input-output")

Une occurrence définit sur quel schéma travaille un traitement, en entrée ou en sortie.

ROLE-IO
définit le role joué par le schéma sur le traitement, description informelle (exemple: consultation, création, destruction, modification, insesrtion, suppression,...)


MEM-AUX
NOM-M
TYPE-M
CLASSE-M
CONNECTE
RES
NOM-RE
TYPE-RE
CLASSE-RE
SERV
NOM-SE
AFFECTE
PERS
NOM-P
DON-P
LOCT
ROLE-LT
UTILISE
TRT-SERV
ROLETP
ROLE-RTP
TRT
NOM-T
TYPE-T
ETAPE-T
DMAJ-T
DEV-T
VERSION-T
PRIVIL-T
LOCF
NOM-LF
DMAJ-LF
ROLE-LF
ROLESP
ROLE-RSP
HIER
I-O
ROLE-IO
FICHIER
NOM-F
CLASSE-F
TYPE-F
SHEMA
NOM-SCH
TYPE-SCH
VERSION-SCH
DEV-SCH
DATE-SCH
PRIVIL-SCH
FTE
STE
TAS
NOM-TAS
ROLE
ROLE-RO
CONNECT-RO
TE
NOM-TE
LIEN-TAS
LIEN-TE
CONC
ATTR
CLASSE-A
NIVEAU-A
TYPE-A
DECOMP
est
ou
COMP 2
ROLE-C2
SENS-C2
NSEQ-C2
IKO
ORDRE-IKO
DOUBLE-IKO
COMP 1
ROLE-C1

# CHAPITRE 5.

# ETUDE DE L'IMPLEMENTATION DU DD.

## 5.1. INTRODUCTION

Nous savons que le schéma conceptuel général peut être vu comme une généralisation de tout schéma conceptuel (cfr. annexe 5.1.). Cette démarche est une vue "par le haut" d'un schéma car le concepteur se place à un niveau d'abstraction supérieur à celui de son propre schéma.

Dans un but d'implémentation, nous allons considérer ce même schéma général comme une vue "par le bas" d'un schéma: c'est-à-dire dans ce cas-ci, une structure d'accueil pour le schéma du dictionnaire de données (que nous appellerons le noyau pour éviter la confusion).

Le schéma général est donc un intermédiaire entre le noyau et le SGBD SQL qui constitue le support à l'UCM. Nous avons choisi cette solution plutôt que l'implémentation directe du noyau en SQL pour permettre l'extensibilité du dictionnaire de données, contrainte de départ.

Ce raisonnement est fortement guidé par la possibilité d'utiliser un outil très puissant du SGBD SQL: les vues ("view" SQL). Créées par une simple requête, elles sont ensuite manipulables pratiquement comme des tables.

La totalité du noyau est implémentée par des vues qui sont les seules structures visibles pour l'utilisateur. L'extensibilité est réalisée par la création ou la modification de celles-ci, ce qui garantit une très grande souplesse car la structure réelle, que nous appelons la structure d'accueil, n'est jamais modifiée. Celle-ci doit répondre à deux objectifs contradictoires:

- être très complète pour permettre une large extension du noyau.
- être très performante, c'est-à-dire limiter le nombre

d'informations à introduire pour une information ajoutée au noyau. Il nous faut donc réaliser un compromis.

Nous allons comparer 3 techniques d'implémentation du schéma général comme structure d'accueil:

- tel quel (c'est-à-dire: ELEMENT, OBJET, RELATION, PROPRIETE, TEXTE)

- par modification du rôle joué par le T.E.:PROPRIETE

- par modification du rôle joué par les T.E.: PROPRIETE et RELATION

Nous justifierons le choix de la troisième technique et nous détaillerons les mécanismes permettant l'extensibilité dans ce cas.

L'annexe 5.1. présente une description complète du schéma général de même que la notion de surclasse et sa traduction dans le modèle E.A.

L'annexe 5.2. rappelle les règles de transformation:
Modèle E.A.- Modèle MAG et les applique au noyau avec les modifications décrites au paragraphe 5.5.

L'annexe 5.3. décrit les tables et vues SQL, qui ont finalement été créés.

## 5.2. ETAPES DE L'IMPLEMENTATION

Les trois techniques d'implémentation présentées suivent le même raisonnement qui peut être divisé en trois étapes.

Etape 1:

Présentation et justification du schéma conceptuel intermédiaire entre le noyau et le SGBD SQL.

Etape 2:

Transformation de ce schéma conceptuel en un schéma MAG compatible SQL. (Les grandes règles de transformation sont rappelées dans l'annexe 5.2).

Etape 3:

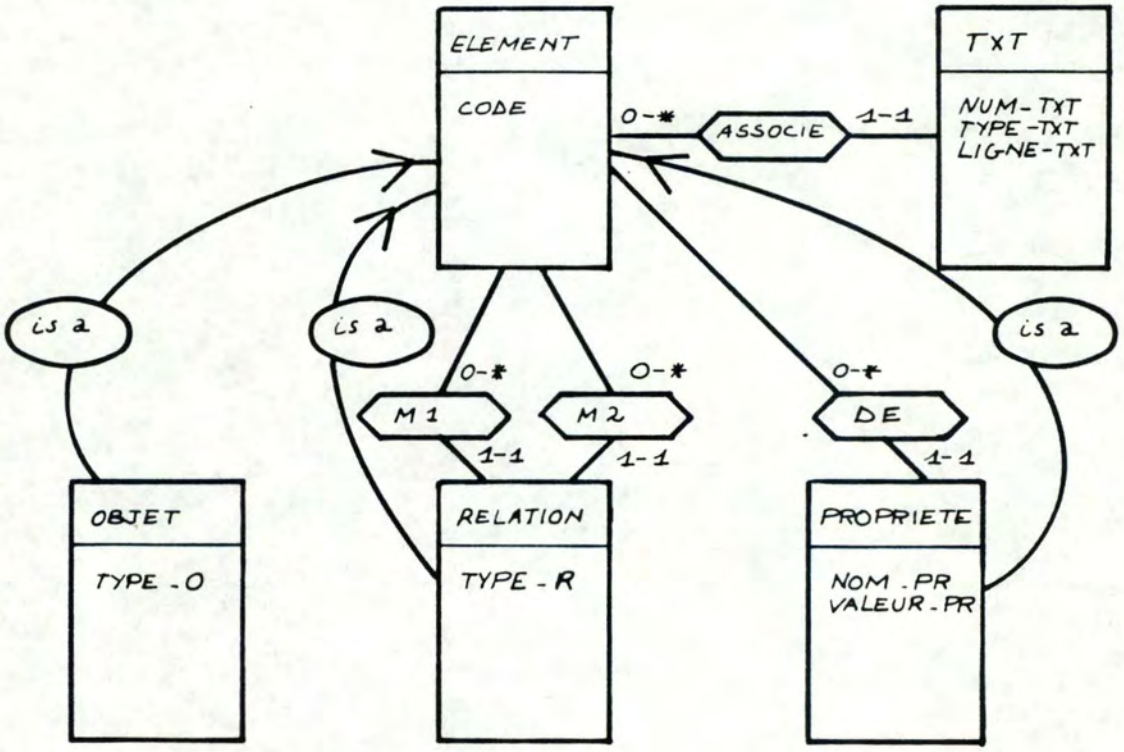
- Création des tables SQL correspondant au schéma MAG obtenu
- Implémentation du noyau, c'est-à-dire création des vues SQL traduisant le schéma conceptuel du noyau.

Le raisonnement complet est présenté pour la première implémentation. Nous n'en présentons que les modifications pour les deux autres techniques.

## 5.3. IMPLEMENTATION PAR LE SCHEMA GENERAL TEL QUEL

### 5.3.1. Etape 1.

Schéma conceptuel (E.A.+ "is a")



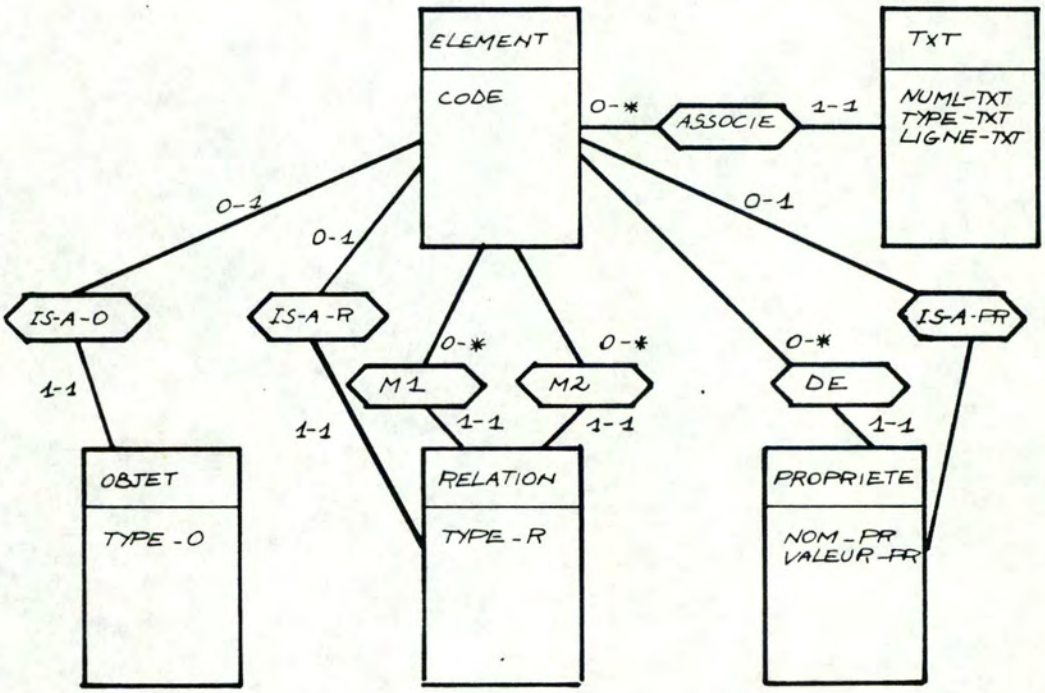

La structure arborescente est traduite par trois nouvelles associations: "is-a-O", "is-a-R", "is-a-PR". Cette solution semble introduire de nouvelles occurrences dans un schéma qui au départ n'est déjà pas très performant (cfr. annexe 5.1. exemple).

Nous la préférons aux deux autres car:

- le nivellement par le bas nécessite la création de 12 nouveaux T.A.:
  - 3 pour remplacer DE,
  - 3 pour remplacer ASSOCIE
  - 6 pour remplacer M1 et M2

- le nivellement par le haut introduit 3 T.A. récursifs et de nombreuses contraintes d'intégrité.

Schéma conceptuel modifié (E.A.)

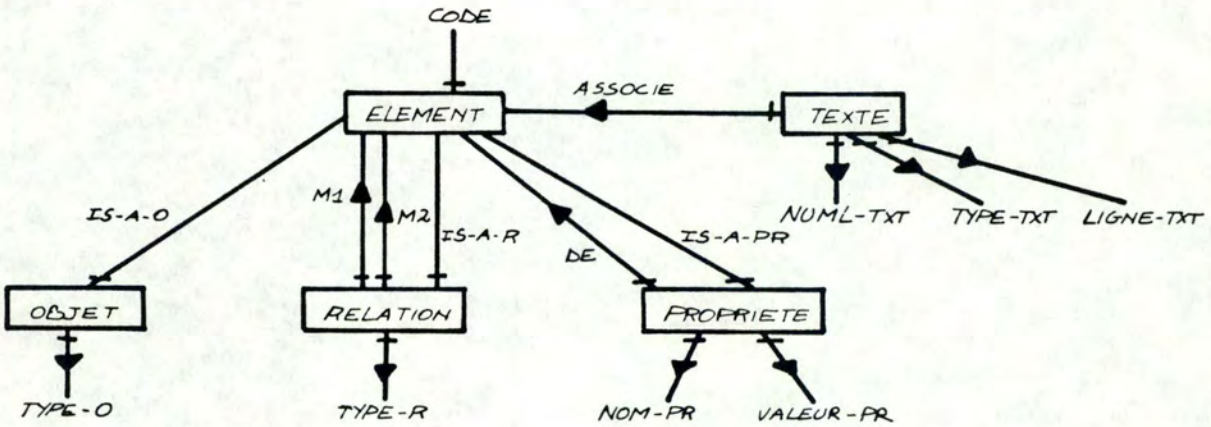



Contrainte d'intégrité: chaque élément est obligatoirement associé soit à un objet par is-a-0, soit à une relation par is-a-R, soit à une propriété par is-a-PR.

### 5.3.2. Etape 2.

Traduisons premièrement le schéma conceptuel modifié en schéma MAG.

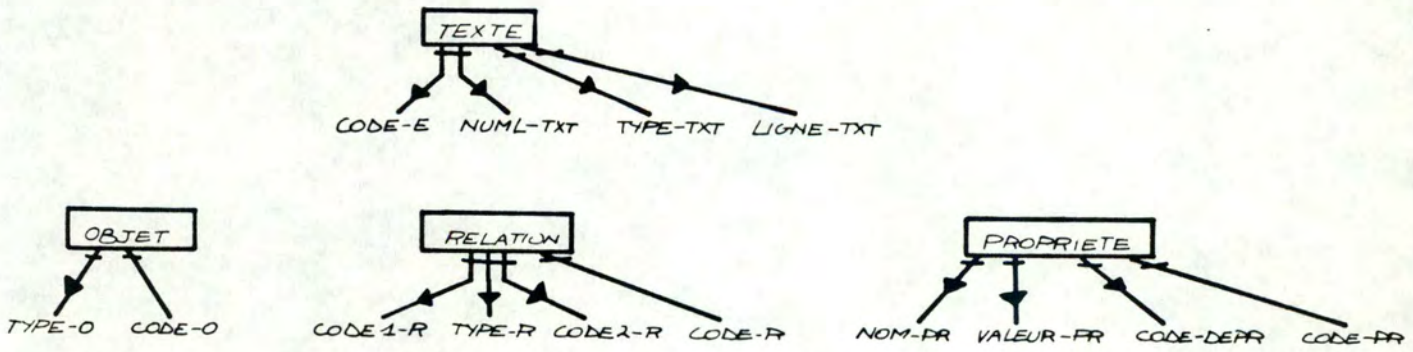
Schéma MAG:




Contrainte d'intégrité: idem.

Modifions ensuite ce schéma pour le rendre compatible avec les contraintes propres aux SGBD relationnels (cfr. annexe 5.2. pour une description de ces contraintes).

Schéma MAG compatible SQL:



Associons la contrainte d'intégrité découverte à l'étape 1, traduisant l'arborescence, aux contraintes liées aux rotations sur le code identifiant. Cela nous permet de supprimer le T.A. ELEMENT sans perte d'information. En utilisant la théorie ensembliste, on peut dire: (CODE) = (CODE-O) U (CODE-R) U (CODE-PR)

Les contraintes de notre dernier schéma seront:

(CODE-E) ≡ (CODE-O) U (CODE-R) U (CODE-PR)
(CODE-O) ≡ (CODE-R) = Ø
(CODE-O) ≡ (CODE-PR) = Ø
(CODE-R) ≡ (CODE-PR) = Ø

### 5.3.3. Etape 3.

#### 5.3.3.1. Les tables

La création des 4 tables est immédiate.
Prenons par exemple la relation:

```
Create table RELATION (CODE-R  varchar (12) not null,
             TYPE-R   varchar (12) not null,
             CODE1-R  varchar (12) not null,
             CODE2-R  varchar (12) not null,)
```

Les deux identifiants sont traduits par :

```
create unique index ID1-R
       on RELATION ( CODE-R asc)

create unique index ID2-R
       on RELATION (TYPE-R asc, CODE1-R asc,
                    CODE2-R asc)
```

#### 5.3.3.2. Les vues:

La requête permettant la sélection des lignes dans les tables pour la création des vues suit les niveaux d'abstraction définis dans le schéma général (cfr. annexe 5.1.).

Nous devons identifier la sous-classe propre à chaque T.E., T.A. et attribut. Pour limiter notre exemple d'implémentation du noyau, créons les vues correspondantes aux types d'entités: TE. et SCHEMA (en limitant ses attributs à NOM-SCH et TYPE-SCH) et au type d'association: STA.

```
1) create view T.E. (NOM-TE) as
   select VALEUR-PR
   from PROPRIETE
   where NOM-PR = 'NOM-TE'and
         CODE-DEPR in (select CODE-O
                       from  OBJET
                       where TYPE-O = 'TE')
```

N.B.: Dans le noyau, le nom d'un attribut l'identifie dans tout le schéma. Nous nous sommes donné cette contrainte dans un souci de clarté et nous ne la considérons pas ici.
Les seules contraintes données dans le schéma général sont que l'attribut soit non répétitif et que le nom identifie l'attribut parmi tous ceux d'un même T.E. ou T.A. (c'est-à-dire la sous-classe), ce qui justifie la seconde contrainte sur la sélection.

```
2) create view STA (CODE-SCH, CODE-TE) as
          select CODE1-R, CODE2-R
            from RELATION
           where TYPE-R = 'STA'
```

N.B.: Nous remarquons ici la nécessité d'établir une convention déterminant quel T.E. sera le membre 1 de la relation et quel autre en sera le membre 2. Etablissons que le T.E. dont le nom apparait le premier dans l'ordre alphabétique est le membre 1. Remarquons que cette convention ne peut s'appliquer si le noyau est exprimé dans un schéma MAG car le concept de type de chemin est unidirectionnel.

Une base de données est toujours supposée cohérente; il est donc inutile d'ajouter des contraintes à notre sélection comme:

```
where CODE1-R in (select CODE-O
                  from OBJET
                  where TYPE-O = 'SCHEMA')

where CODE2-R in (select CODE-O
                  from OBJET
                  where TYPE-O = 'TE')
```

```
3) create view SCHEMA (NOM-SCH, TYPE-SCH) as
        select P1.VALEUR, P2.VALEUR
          from PROPRIETE P1, PROPRIETE P2
         where P1.NOM-PR. = 'NOM-SCH' and
               P2.NOM-PR = 'TYPE-SCH' and
               P1.CODE-DEPR in
              (select CODE-O
                 from OBJET
                where TYPE-O = 'SCHEMA') and
               P2.CODE-DEPR in
              (select CODE-O
               from OBJET
               where TYPE-O ='SCHEMA')
```

N.B.: La sélection est ici la même que pour la vue T.E., doublée pour considérer les deux attributs de schéma.

### 5.3.4. EVALUATION.

Les possibilités d'extension du schéma général tel quel sont pratiquement illimitées: création d'un nouveau T.E. avec des attributs quelconques, création d'un nouveau T.A. sur des T.E. quelconques,...

Il impose cependant une limite importante: le type de valeur des attributs est unique pour tous les attributs définis dans le noyau. En pratique, ce ne peut être qu'une chaine variable de caractères, ce qui empêche d'effectuer toutes les manipulations que SQL permet pour les valeurs numériques (+, *, AVG,...) et consomme de l'espace-mémoire.

Ses performances sont inacceptables en nombre d'occurrences créées (cfr. annexe5.1.) ce qui nous conduit à rejeter le schéma général tel quel et à rechercher une solution plus simple.

## 5.4. IMPLEMENTATION PAR LE SCHEMA GENERAL EN MODIFIANT LE ROLE JOUE PAR LE T.E.: PROPRIETE

### 5.4.1. Etape 1.

Le principal inconvénient de la première implémentation est la lourdeur de manipulation des valeurs d'attributs.

Dans une première phase, supprimons le T.E. PROPRIETE; nous insérons le noyau dans le schéma réduit en réalisant un nivellement par le haut, chaque T.E. et chaque T.A. étant une sous-classe de OBJET et RELATION.

Au niveau de la conception, ceci nous oblige à ajouter à ces 2 T.E. tous les attributs des T.E. et T.A. du niveau inférieur avec des contraintes d'exclusion sur le type. Puisque notre optique est l'implémentation et que les vues SQL permettent de modifier les noms des colonnes, il nous suffit de considérer le nombre minimum d'attributs pour les T.E.: OBJET et RELATION. Leurs valeurs auront une signification différente dans chaque sous-classe considérée. Ceci nous permet en outre de distinguer différents types de valeurs aux colonnes des 2 T.E.

Nous ne détaillerons pas ici la matrice des correspondances entre noms de colonnes. Celle-ci est faite dans la troisième technique d'implémentation, celle qui a été adoptée. Nous supposerons que:

- OBJET possède 9 attributs: CODE-O, TYPE-O, C3-O,C4-O, C5-O, C6-O, C7-O, C8-O, C9-O.
- RELATION possède 5 attributs:CODE-R, TYPE-R, C3-R,C4-R, C5-R.

Dans une deuxième phase, considérons les possibilités d'extension du noyau. Une limite est la création de T.E. ou T.A. qui ont un plus grand nombre d'attributs.

Nous pallions cet inconvénient en réintroduisant le T.E.: PROPRIETE et le T.A.: DE. Chacune de ses occurrences correspondra à un attribut et sa valeur pour un T.E. ou un T.A. ajouté ou modifié et qui possèdera plus de 7 attributs.
La liaison de chacun de ces attributs supplémentaires à son T.E. ou T.A. est réalisée comme précédemment. Ceci limite la lourde manipulation de PROPRIETE à des cas exceptionnels.

### 5.4.2. Etape 2.

La forme du schéma général étant seulement modifiée par l'ajoute des attributs: C3-O, C4-O, C5-O, C6-O, C7-O, C8-O, C9-O et C3-R, C4-R, C5-R et la suppression du "is a" de propriété vers élément, la transformation en schéma MAG compatible SQL est similaire à celle du premier schéma.

### 5.4.3. Etape 3.

#### 5.4.3.1. Les tables.

Leur création est immédiate.

#### 5.4.3.2. Les vues.

Sans modifier la correspondance définie entre le noyau et le schéma général, il est possible de simplifier la structure visible pour l'utilisateur c.à.d. les vues.

Plutôt que de créer des vues reflétant exactement la correspondance entre les 2 schémas, présentons-les comme une simulation de l'implémentation directe du noyau en SQL, c.à.d. comme si ce dernier était transformé en schéma MAG, compatible relationnel, en effectuant des rotations. On considère évidemment que chaque type d'article obtenu possède un item identifiant: son code. Cet item sert pour les rotations.

De cette façon, nous ne créons pas de vue pour les T.A. comme LIEN-TE, STA, CONC, ..., possédant une connectivité (1-1).

Créons les vues pour le même exemple que précédemment.Nous supposons que les attributs de TE. et SCHEMA sont respectivement le premier et les 2 premiers attributs libres de OBJET.

```
1) create view SCHEMA (CODE-SCH, NOM-SCH, TYPE-SCH) as
          select CODE-O, C3-O, C4-O
          from OBJET
          whère TYPE-O = 'SCHEMA'

2) create view TE (CODE-TE, NOM-TE, CODE-SCHTE) as
          select CODE-O, C3-O, C4-O, CODE 1-R
          from OBJET, RELATION
          where TYPE-O = 'TE'   and
                TYPE-R = 'LIEN' and
                CODE-O = CODE2-R
```

### 5.4.4. Evaluation.

Les possibilités d'extension sont les mêmes que précédemment grâce au rôle joué par le T.E. PROPRIETE.

Les types de valeurs des attributs sont prédéfinis mais peuvent être choisis parmi les possibilités offertes par le SQL, notamment pour les valeurs numériques.

Le nombre d'occurrences enregistrées est fortement diminué, ce qui améliore sensiblement les performances malgré l'apparition de lignes dont certaines valeurs sont vides ("null value"), surtout pour les T.A. sans attribut du noyau.

La manipulation du code identifiant, pour toute occurrence d'un type quelconque, reste très lourde et n'apporte aucune information.

## 5.5.IMPLEMENTATION.PAR LE SCHEMA GENERAL EN MODIFIANT LE ROLE JOUE PAR LES T.E.: PROPRIETE ET RELATION

### 5.5.1. Etape 1

Jusqu'à présent nous avons considéré que le noyau était modélisé dans le formalisme E.A. C'est de cette façon que toute la conception a été réalisée. Par contre, le schéma général permet d'implémenter beaucoup plus directement un modèle binaire. En effet, la RELATION relie 2 ELEMENTS (non nécessairement distincts). Nous savons qu'un T.A. ternaire ne peut pas être représenté dans le schéma général (cfr. annexe 5.1.).

Le modèle MAG est binaire et la notion de chemin d'accès s'identifie à la RELATION. Il faut cependant remarquer que nous nous limitons au schéma des accès possibles c'est-à-dire que tout chemin possède un inverse. Dans notre cas, deux chemins inverses l'un de l'autre sont représentés par la même RELATION. Celle-ci ne considérera donc pas la direction d'un chemin. Les conventions adoptées pour les T.A. M1 et M2 restent donc celles décrites précédemment (ordre alphabétique).

Pourquoi adopter cette convention ?
Comme le SGBD est relationnel, il implémente toujours ces liaisons (M1 et M2) par des jointures sur des tables; une jointure est, par définition, bidirectionnelle. Nous pouvons donc conserver cette généralité et implémenter directement le schéma des accès possibles du noyau. L'annexe 5.2. rappelle les grandes règles de transformation E.A - MAG et les applique au schéma du noyau.

La RELATION ainsi définie ne possèdera jamais de PROPRIETE. Il n'est pas utile non plus de permettre la liaison entre deux RELATIONS ni d'associer un TEXTE à une RELATION. En effet, toute description se rapporte à un OBJET, c.à.d. dans notre cas un article. Nous éliminons donc la RELATION de l'arborescence.

Nous faisons de même pour la PROPRIETE ce qui ne nous permet plus de représenter les items décomposables (car ils sont seulement reliés à leur type d'article ). Ceci n'est pas une contrainte pour nous car tous les items du noyau sont élémentaires.

Notre schéma définitif réduit ramène l'arborescence à la surclasse ELEMENT avec la seule classe OBJET. Nous l'éliminerons par nivellement par le haut (convenons d'appeler OBJET le T.E. restant pour plus de clarté).

Le dernier problème concerne les attributs :
Le T.E. OBJET hérite de tous les attributs correspondant aux items des types d'articles du noyau (nivellement par le haut). Le nombre d'attributs est réduit en suivant le même raisonnement que pour la première modification (cfr. 5.4.1.).

Pour éliminer le code qui identifie chaque élément, nous devons faire un choix. La plupart des T.E. du noyau sont identifiés par leur nom. Si nous généralisons ce principe à chaque type d'article du schéma MAG du noyau, nous pouvons identifier un OBJET par son nom et son type. Nous avons adopté cette solution qui représente les inconvénients d'ajouter un nom à certains T.A. du noyau et surtout de donner un nom unique à chaque ATTRIBUT décrit par le dictionnaire. Nous verrons dans les chapitres 6 et 7 comment est pallié cet inconvénient.

Le T.E. RELATION ne conserve aucun attribut propre, ce qui ne nous permet pas de distinguer deux types de chemins distincts reliant deux mêmes types d'articles (cas exceptionnel en pratique et ne se présentant pas dans le noyau). Chaque RELATION sera identifiée par les deux OBJETS sur lesquels elle porte.

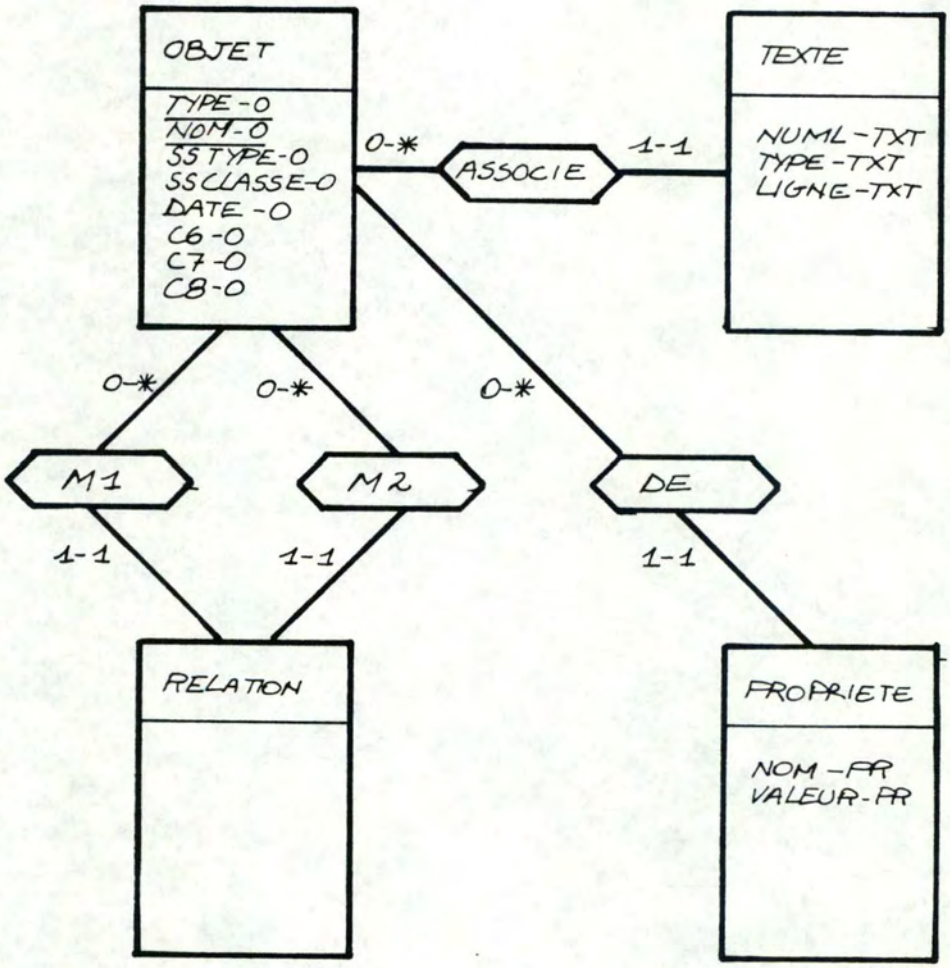
Le T.E. propriété est identifié par son nom, sa valeur et l'identifiant de l'objet sur lequel il porte car, rappelons-le, il est :
- simple (cfr annexe 5.1.2.6.)
- élémentaire (cfr ci-dessus)

Nous obtenons la matrice de correspondance suivante en conservant des noms significatifs là où c'est possible.
Schéma conceptuel définitif
(voir page suivante)

ATTRIBUTS DE L'OBJET

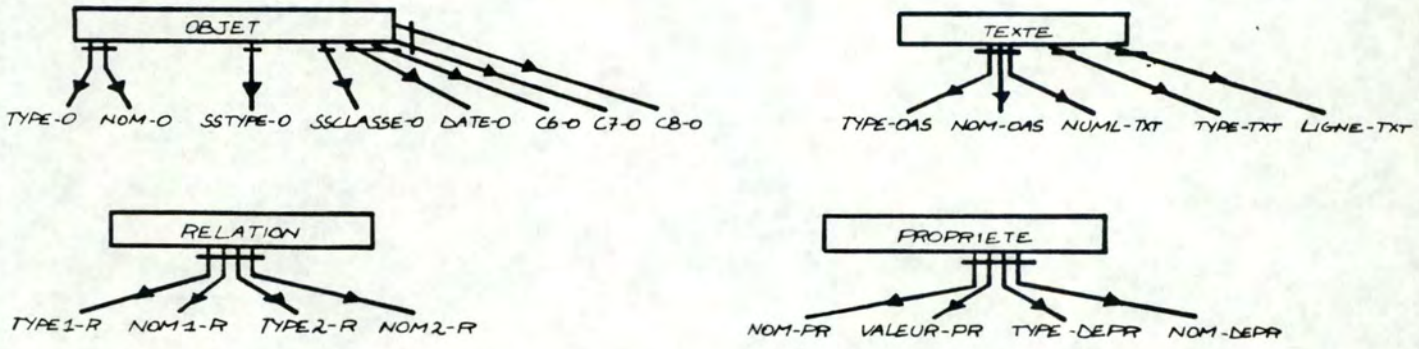
| nom de l'attribut de l'objet | nom du T.E. ou du T.A. du noyau | NOM-O | SS-TYPE-O | SS-CLASSE-O | DATE-O | C6-O | C7-O | C8-O |
|---|---|---|---|---|---|---|---|---|
| nom des attributs correspondants des T.E. du noyau | mem-aux | nom-m | type-m | classe-m | - | - | - | - |
| | res | nom-re | type-re | classe-re | - | - | - | - |
| | serv | nom-se | - | - | - | - | - | - |
| | pers | nom-p | - | - | ddn-p | - | - | - |
| | trt | nom-t | type-t | étape-t | dmaj-t | dev-t | version-t | privil-t |
| | fichier | nom-f | type-f | classe-f | - | - | - | - |
| | schéma | nom-sch | type-sch | - | date-sch | dev-sch | version-sch | privil-sch |
| | tas | nom-tas | - | - | - | - | - | - |
| | te | nom-te | - | - | - | - | - | - |
| | iko | nom-iko | type-iko | - | - | ordre-iko | - | doubles-iko |
| | attr | nom-a | type-a | classe-a | - | - | - | - |
| nom des attributs correspondants des T.A. du noyau | loct-t | nom-lt | - | - | - | role-lt | - | - |
| | role-tp | nom-tp | - | - | - | role-tp | - | - |
| | locf | nom-lf | - | - | dmaj-lf | role-lf | - | - |
| | rolesp | nom-sp | - | - | - | role-sp | - | - |
| | i-o | nom-io | - | - | - | role-io | - | - |
| | role | nom-ro | - | - | - | role-ro | - | connect-ro |
| | comp1 | nom-c1 | - | - | - | role-c1 | - | sens-c1 |
| | comp2 | nom-c2 | - | - | - | role-c2 | - | sens-c2 |
| type de valeur de l'attribut | varchar(12) | varchar(12) | varchar(12) | varchar(12) | integer | varchar(12) | smallint | varchar(2) |

Schéma conceptuel définitif



### 5.5.2. Etape 2

Schéma MAG compatible SQL




Les transformations ajoutent des contraintes d'intégrité que nous exprimons en utilisant la théorie ensembliste :

(TYPE-OAS) $\subset$ (TYPE-O)
(NOM-OAS) $\subset$ (NOM-O)
(TYPE1-R) $\subset$ (TYPE-O)
(NOM1-R) $\subset$ (NOM-O)
(TYPE2-R) $\subset$ (TYPE-O)
(NOM2-R) $\subset$ (NOM-O)
(TYPE-DEPR) $\subset$ (TYPE-O)
(NOM-DEPR) $\subset$ (NOM-O)

### 5.5.3..Etape 3

La création des tables et vues est complètement décrite dans l'annexe 5.3. Comme pour l'évolution précédente, la création des tables est immédiate et les vues créées simulent l'implémentation directe du noyau en SQL.

Résumons la double correspondance :
schéma MAG du noyau - tables SQL - vues SQL

- articles avec leurs items
  table OBJET
  vues correspondant aux types

- chemins (+inverses) de type M-N
  table RELATION
  vues correspondant aux types

- chemins (+ inverses) de type 1-N
  table RELATION
  ajoute d'une colonne dans les vues correspondant aux types d'articles concernés.

Parmi les vues créées, certaines simulent la rotation de deux items, ce sont celles qui correspondaient à des T.A. avec attributs dans le modèle E.A du noyau. Leur création est plus délicate car elles possèdent des colonnes issues de deux relations différentes. (cfr annexe. 5.3 : vues LOCT, ROLETP, ROLESP...)

Une seule exception est faite au principe de création : la vue ATTR. La contrainte d'exclusion sur les chemins LIEN-TE et LIEN-TAS est traduite par l'ajoute de deux colonnes dans la vue :

- NOM-LIEN identifiant soit un TAS, soit un T.E.
  (sans permettre de "null value")
- TYPE-LIEN précisant s'il s'agit d'un TAS ou d'un T.E.
  (sans permettre de "null value")

### 5.5.4 Evaluation

Par rapport à la seconde technique d'implémentation, on observe les modifications suivantes:

- Les possibilités d'extensions sont limitées pour les nouveaux items non prévus dans la table OBJET : ils seront élémentaires (aucun inconvénient en pratique).

- Le nombre d'occurrences à légérement augmenté mais la table RELATION possède quatre colonnes au lieu de sept.

- Le nombre de valeurs vides a diminué (aucune dans la table RELATION et un moins grand nombre dans la table OBJET).

- La lourde manipulation du code identifiant a disparu.

- Les principaux inconvénients sont l'introduction d'un item non significatif dans les types d'articles issus des T.A. avec attributs et surtout l'identifiant unique et formé d'un seul item (le nom généralement) pour chaque type d'article du noyau ( y compris les extensions futures).

## 5.6. MECANISMES PERMETTANT L'EXTENSIBILITE.

### 5.6.1. Possibilités offertes.

Différentes possibilités sont offertes à l'utilisateur désirant étendre son dictionnaire de données pour représenter de nouvelles notions. En considérant le schéma MAG du noyau (cfr annnexe 5.2.), il peut :

- ajouter un type de chemin quelconque entre deux types d'articles si ceux-ci ne sont pas déjà reliés par un autre type de chemin. (Ceci correspond dans le schéma conceptuel à ajouter un T.A. sans attributs entre deux T.E., entre un T.E. et un T.A. avec attributs ou entre deux T.A. avec attributs.)

- ajouter un type d'article avec un nombre quelconque d'items simples et élémentaires, dont un sera obligatoire et identifiera le type d'article . Ce dernier peut ensuite être relié à d'autres en suivant la restriction ci-dessus.

- ajouter un item au type d'article dont il dépend.

- ajouter un OBJET d'un type particulier, le type MAPPING, qui permet de relier un premier ensemble d'occurrences à un second, celles-ci étant choisies indistinctement parmi tous les types d'articles du noyau (qu'ils soient prédéfinis ou issus d'une extension particulière.)(cfr. chap 3).

Toutes les extensions définies font ensuite partie intégrante du noyau dans lequel tout type d'article ou de chemin peut-être modifié ou supprimé.

Nous allons prendre un exemple pour chaque cas. Nous définirons la vue créée et les occurrences ajoutées aux tables qui implémentent la structure intermédiaire. Nous ne préciserons pas la sémantique liée à nos exemples. Ceci est un des objectifs des chapitres 4 et 7.(sauf pour le troisième cas qui représente la simple ajoute d'une colonne à une vue). La suppression d'un type entraine la simple destruction d'une vue et des occurrences correspondantes dans les tables. La modification d'un type entraine la destruction et la création d'une nouvelle vue et la mise à jour des occurrences correspondantes dans les tables.

### 5.6.2. Ajouter un type de chemin.

Définissons par exemple le type de chemin TRT-LOCF entre les types d'articles TRT et LOCF. En suivant les principes de définition des vues, nous procédons comme suit :

- s'il a une connectivité : M-N ou 1-N (sans contrainte d'existance du côté 1- ), nous créons une nouvelle vue sur RELATION qui reprend les noms du traitement et du fichier physique.

- s'il a une connectivité 1-N (avec contrainte d'existence du côté 1- ), nous ajoutons à la vue correspondant à ce type d'article une colonne représentant le nom du second type d'article.

Exemple d'occurrences : prenons le cas du type de chemin M-N sans contraintes. Nous avons au départ :

- LOCF 1, LOCF 2
- TRT 1, TRT 2

c'est-à-dire dans la table OBJET

OBJET

| TYPE-O | NOM-O | |
|---|---|---|
| TRT | TRT1 | |
| TRT | TRT2 | |
| LOCF | LOCF1 | |
| LOCF | LOCF2 | |

- relions TRT 1 à LOCF 1 et LOCF 2
  et TRT 2 à LOCF 2

nous obtenons dans la table RELATION

| TYPE1-R | NOM1-R | TYPE2-R | NOM2-R |
|---|---|---|---|
| TRT | TRT1 | LOCF | LOCF1 |
| TRT | TRT1 | LOCF | LOCF2 |
| TRT | TRT2 | LOCF | LOCF2 |

### 5.6.3 Ajouter un type d'article.

Ajoutons par exemple le type d'article MV (machine virtuelle) avec ses items :

NOM-MV identifiant (12 caractères)
DATECR-MV (date de création) (entier)

Nous créons une nouvelle vue :

```
create view   MV (NOM-MV, DATECR-MV)
       select NOM-O, DATE-O
       from   OBJET
       where  TYPE-O = 'MV'
```

Exemple d'occurrences :

Insérons deux MV dans notre base de donnée :
- MV 1 créée le 1/08/85
- MV 2 créée le 3/08/85

Nous aurons dans la table OBJET:

OBJET

| TYPE-O | NOM-O | SSTYPE-O | SSCLASSE-O | DATE-O | C6-O | C7-O | C8-O |
|---|---|---|---|---|---|---|---|
| MV | MV1 | - | - | 10885 | - | - | - |
| MV | MV2 | - | - | 30885 | - | - | - |

Remarque :

Le cas exceptionnel de création d'un type d'article possédant plus de 7 items fait intervenir la table PROPRIETE. Un schéma d'occurrences explicitant ce cas est développé dans l'annexe 5.1.

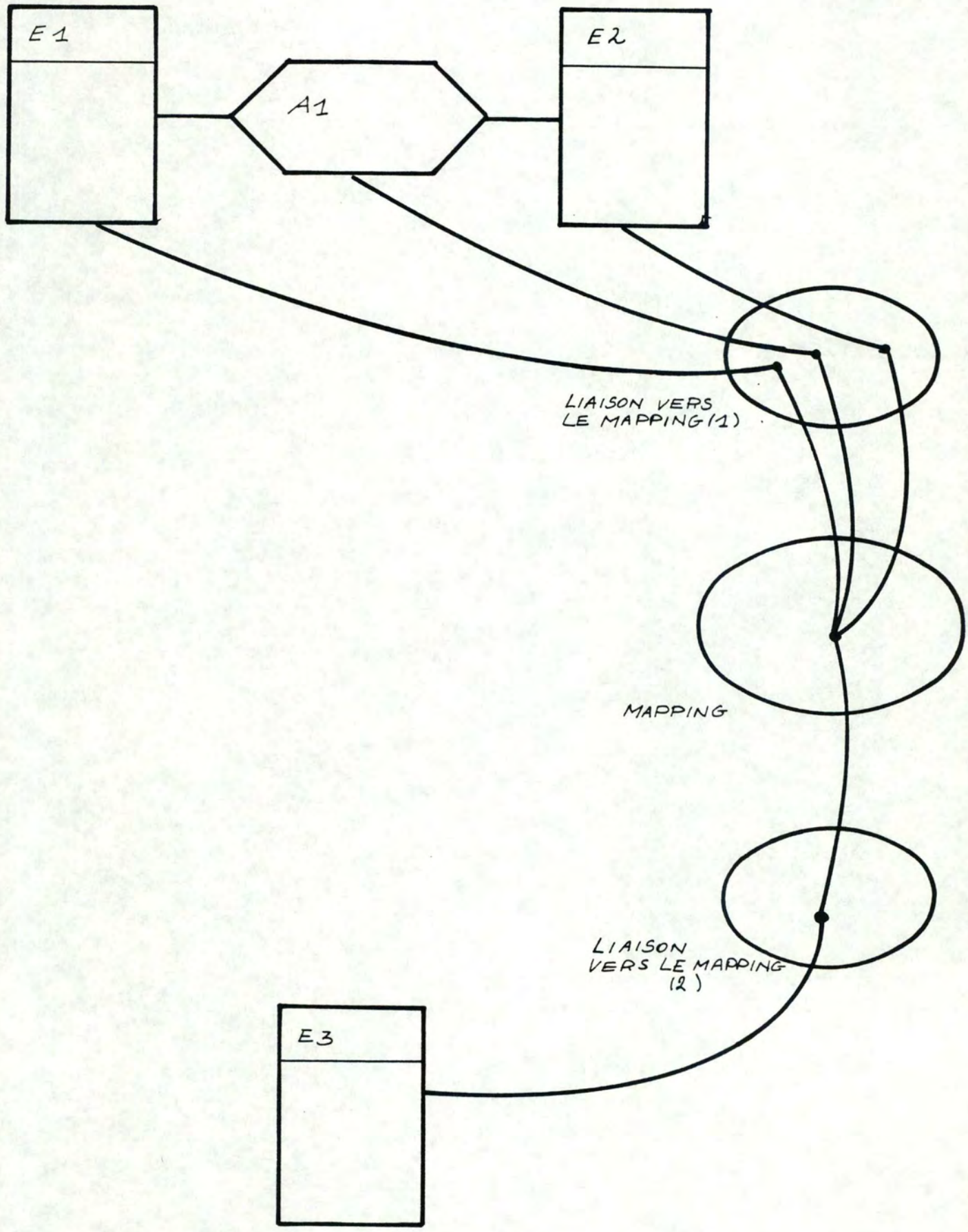
### 5.6.4. Ajouter un objet de type MAPPING.

#### 5.6.4.1. Exemple

Pour comprendre l'utilité de ce nouveau type, prenons un exemple concrèt: celui d'un schéma conceptuel décrit à l'aide du dictionnaire de données. Dans ce schéma, appelons le SCH1, nous avons notamment défini deux T.E.: E1 et E2, et un T.A. les reliant, A1. (Le rôle de chaque T.E. dans le T.A. ne nous intéresse pas ici).

Dans une version ultérieure de notre schéma, appelons-le SCH2, nous préférons simplifier cette structure en la représentant par un seul T.E, appelons-le E3. Comme nous désirons conserver la description des différentes étapes de la conception, nous ne supprimons pas SCH1 et nous représentons la correspondance entre nos deux structures par un MAPPING.

Schématiquement, cela donne: (voir page suivante)


E 1
A1
E 2
LIAISON VERS
LE MAPPING (1)
MAPPING
LIAISON
VERS LE MAPPING
(2)
E 3

### 5.6.4.2. Première technique d'implémentation

Le schéma d'occurrences défini ci-dessus suppose la création d'un T.E.:MAPPING, et de 2 T.A.: LIEN1-M et LIEN2-M; chacun de ceux-ci est logiquement relié d'une part au T.E. MAPPING et d'autre part à chaque T.E. du noyau et de ses extensions éventuelles. En d'autres mots, nous ne créons pas 2 T.A. mais plutot 2 ensembles de T.A..

L'implémentation de ces 2 éléments dans notre structure d'accueil n'est pas aussi simple qu'elle n'y paraît. Si d'une part le T.E. MAPPING peut être representé par un OBJET d'un nouveau type, d'autre part, les 2 ensembles de T.A. ne peuvent être des RELATIONS bien que nous avions convenu de traduire de cette façon tout T.A. sans attribut.

En effet, cette implémentation ne nous permet pas de distinguer, dans chaque cas, le premier ensemble de RELATIONS du second. Nous avons précisé (5.5.1.) l'impossibilité d'associer 2 mêmes T.E. par 2 T.A. distincts et sans attribut (c.à.d. par 2 types de chemins.).

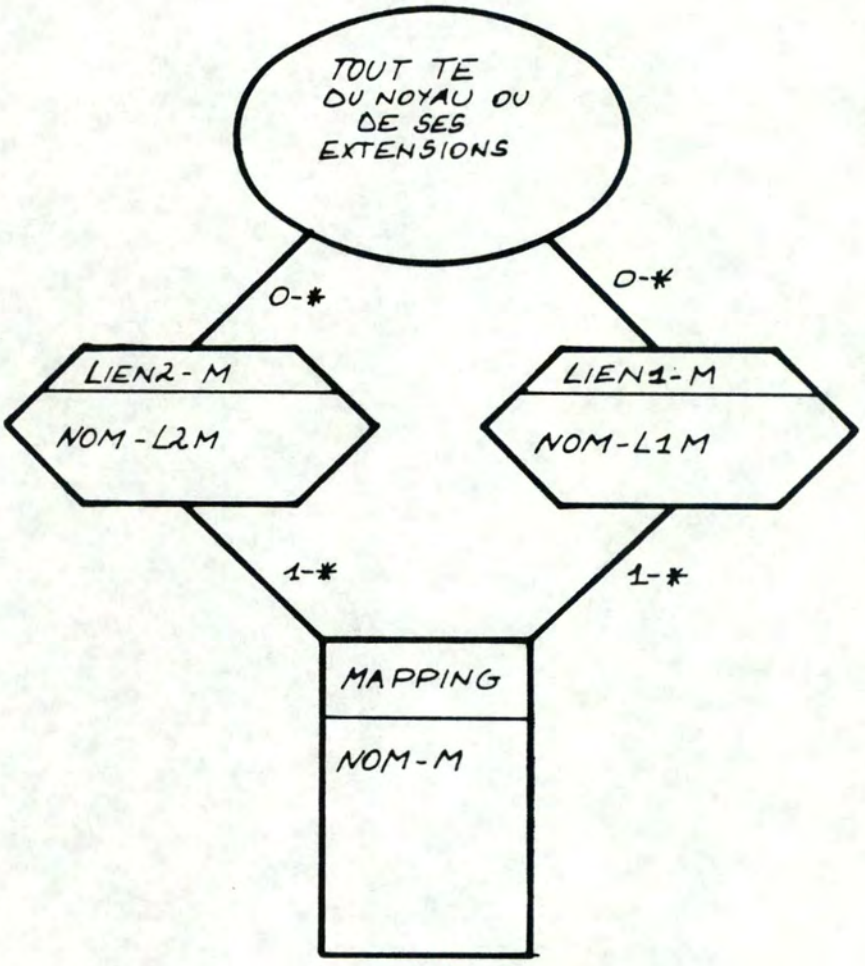
Nous allons donc les implémenter également par des OBJETS, ce qui nous oblige à leur donner un attribut propre et identifiant: leur nom.

Bien que non significative, cet ajout est ici obligatoire. Chacun de ces OBJETS sera relié par des RELATIONS, d'une part au MAPPING et d'autre part à chaque autre T.E. du schéma.

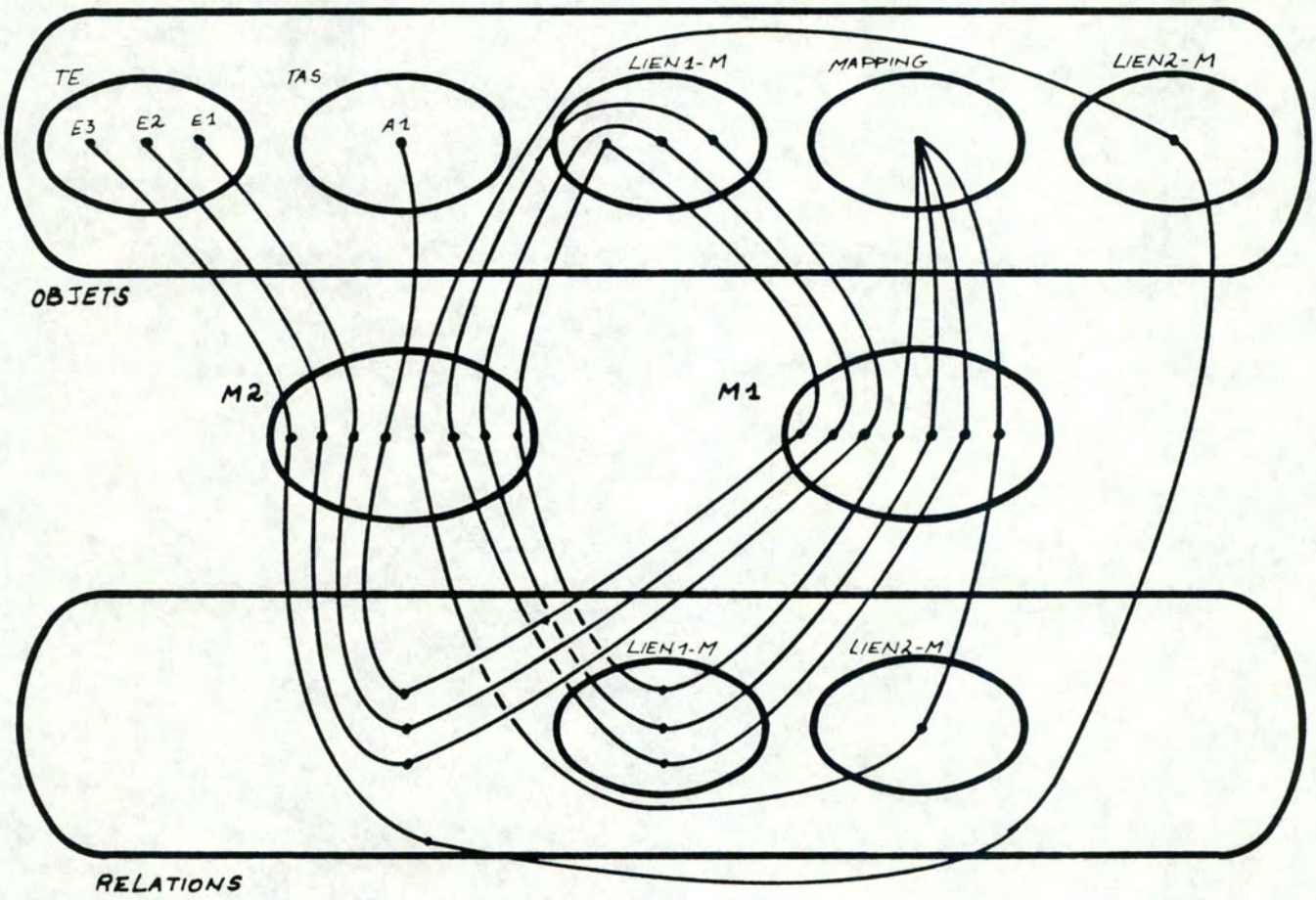
Adoptons les conventions suivantes pour les membres 1 et 2 de chacune de ces RELATIONS, respectivement: M1 et M2:

- MAPPING - LIEN1-M
- MAPPING - LIEN2-M
- LIEN1-M - autre T.E.
- LIEN2-M - autre T.E.

Nous obtenons le schéma conceptuel suivant:
(voir page suivante)




Le schéma d'occurrences correspondant à notre exemple est traduit dans la structure d'accueil comme suit: (voir page suivante)


TE
E3
E2
E1
TAS
A1
LIEN1-M
MAPPING
LIEN2-M
OBJETS
M2
M1
LIEN1-M
LIEN2-M
RELATIONS

Nous remarquons que, bien que très clair dans les principes, la technique d'implémentation des OBJETS et RELATIONS correspondant à un MAPPING est assez lourde. Dans le but de simplifier la représentation de cette correspondance nous présentons une seconde méthode d'implémentation.

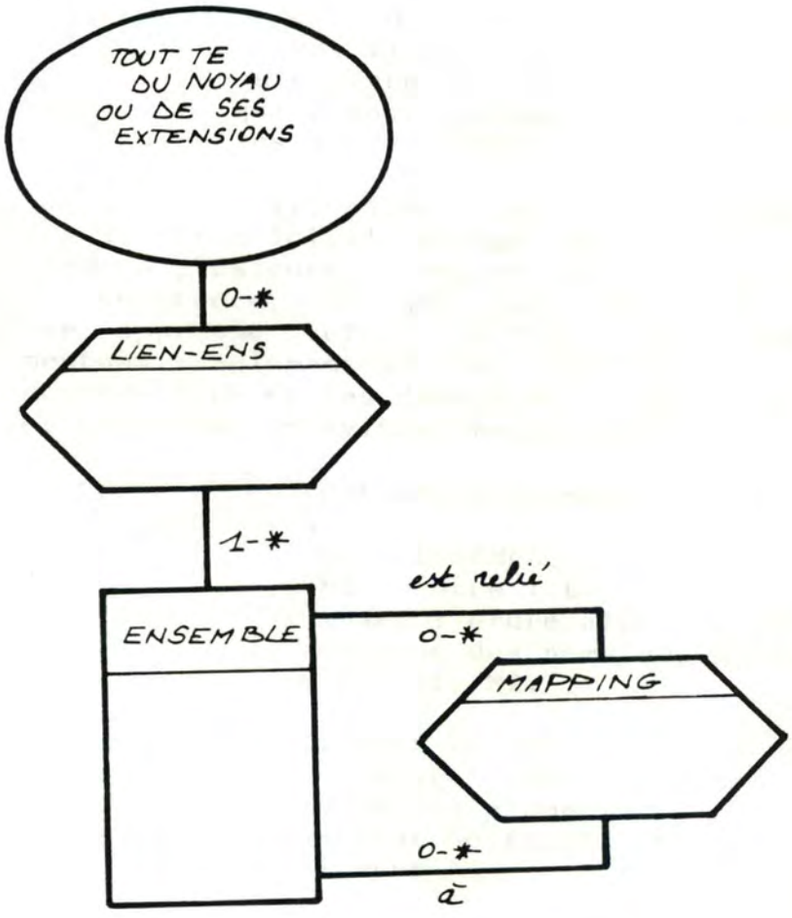
### 5.6.4.3. Seconde technique d'implémentation.

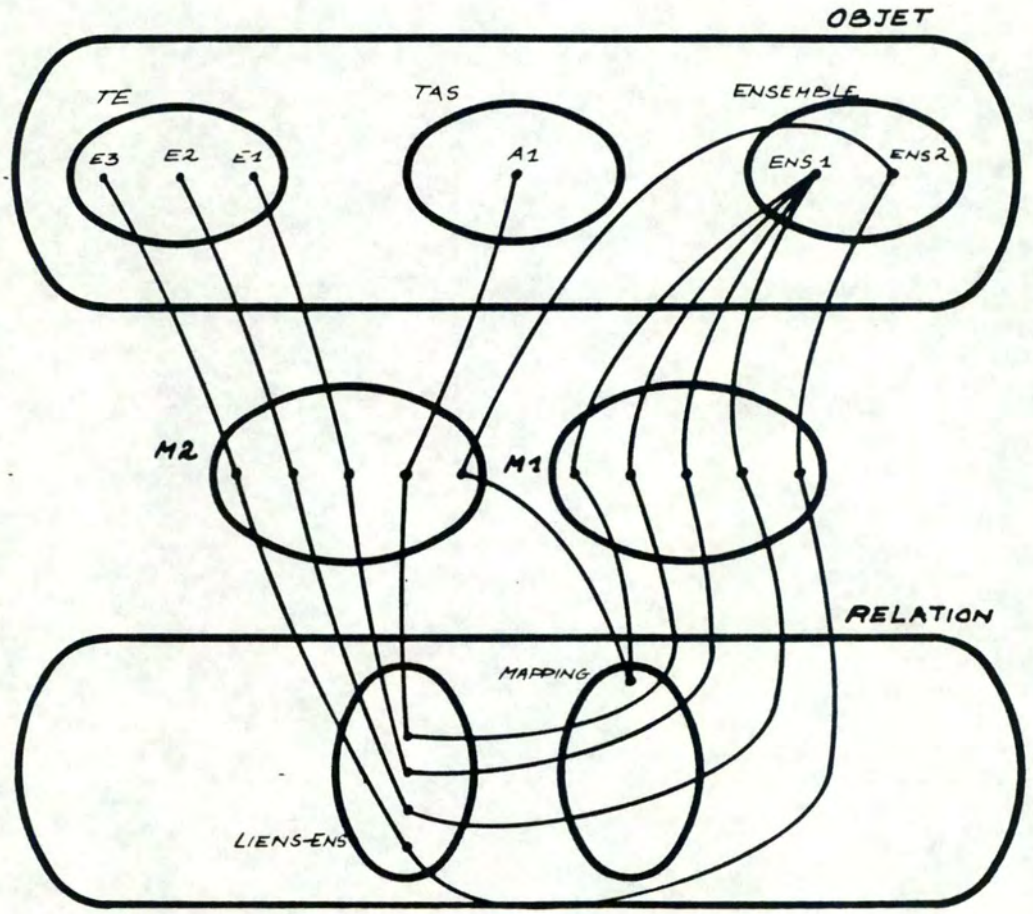
Au delà d'une réflexion sur l'implémentation, considérons les concepts liés à un MAPPING. L'idée principale en est la liaison entre deux ensembles d'occurrences.

Dans notre première approche, nous avons privilégié le MAPPING pour ensuite considérer les deux ensembles en présence. Iversément, nous pouvons d'abord modéliser la notion d'ensemble pour ensuite les relier entre eux. Nous allons voir que le modèle résultant est plus simple à implémenter bien qu'il traduise moins clairement le principe du MAPPING.

Créons simplement le T.E.: ENSEMBLE dont chaque occurrence représente le rassemblement de une ou plusieures occurrences d'un type quelconque du noyau (et de ses extensions). Il sera relié par plusieurs T.A. sans attributs à chaque T.E. dont il regroupe les occurrences. Nous représentons ensuite les liens entre plusieurs ENSEMBLES par un T.A. récursif que nous appellons logiquement MAPPING.

Ceci est représenté par le schéma conceptuel suivant: (voir page suivante)


TOUT TE
DU NOYAU
OU DE SES
EXTENSIONS
0-*
LIEN-ENS
1-*
est relié
ENSEMBLE
0-*
MAPPING
0-*
à


OBJET
TE
E3
E2
E1
TAS
A1
ENSEMBLE
ENS1
ENS2
M2
M1
RELATION
MAPPING
LIENS-ENS

# CHAPITRE 6

# MANIPULATION DU DD.

Comme toute BD, le DD est destiné à etre manipulé par différents utilisateurs. Nous avons vu la nécessité, dans le cadre de ce travail, de nous limiter au rôle passif du dictionnaire. Nous allons passer en revue les différents outils soit directement disponibles, soit à construire pour les deux types d'utilisation courants d'un DD: la modification d'un contenu et la production d'informations.

## 6.1. MODIFICATION DU CONTENU.

### 6.1.1. Manipulation directe des tables

C'est le moyen le plus immédiat de mise-à-jour: l'insertion, la suppression et la modification de lignes dans les quatre tables: OBJET, RELATION, PROPRIETE, TEXTE. Le SQL étant un SGBD évolué, il possède un langage de manipulation assez riche. ("insert", "input", "delete", "update",...). Cependant, ceci est tout à fait contraire au but poursuivi dans ce travail. L'étape d'implémentation (cfr. chapitre 5) a introduit cette structure pour des raisons de performances et d'extensibilité. Elle ne doit pas être connue de l'utilisateur courant qui considère le dictionnaire comme un outil de travail aussi simple à utiliser que cela est possible.

L'utilisation directe des tables sera donc idéalement réservée aux concepteurs des outils de manipulation plus évolués ainsi qu'à ces outils eux-mêmes que nous passons précisément en revue.

### 6.1.2. Manipulation directe des vues.

Pour distinguer les possibilités que le SGBD fournit, considérons à nouveau la notion de vue (introduite au chapitre 5).
La création d'une vue correspond à une requête sur une ou plusieurs tables. Une requête quelconque faite sur une vue est logiquement identique à la superposition de celle-là à celle de la création de la vue, ces deux requêtes combinées étant effectuées sur la (ou les) table(s) de départ. De ceci découle que en SQL toute requête de mise-à-jour d'une vue est logiquement subordonnée à plusieurs restrictions sur la création de celle-ci:

- Aucune commande d'insertion ne peut être faite sur les colonnes de vues ne dérivant pas directement de celles de tables; par exemple: col 1 - col 2, col 3 - 50. Ceci ne nous concerne pas car aucune de nos vues ne possède une telle définition (cfr. annexe 5.3.).

- Aucune modification ("insert", "delete", "update") n'est permise sur une vue définie par les opérations: jointure, fonctions prédéfinies comme "avg",...
Ceci est pour nous une sérieuse restriction: un grand nombre de nos vues sont créés par la jointure des deux tables: OBJET et RELATION (cfr. annexe 5.3.).

Une restriction plus générale interdit de supprimer la valeur d'une colonne pour laquelle la valeur nulle n'est pas permise sans supprimer la ligne entière. Or, nous n'avons jamais la possibilité de supprimer le type de l'OBJET, cette colonne n'apparaissant dans aucune vue; par contre, le nom de l'objet est une colonne obligatoire.

En définitive, il ne nous est jamais permis de modifier directement une vue. Même si cette dernière restriction était levée, toute suppression introduirait des occurrences "vides" dans OBJET et RELATION (définis seulement par leur type) dont l'utilisation entrainerait de très nombreux désagréments.

Un utilisateur sera donc obligé de manipuler des outils plus évolués, ceux-ci utilisant directement les quatre tables.
Nous allons les distinguer.

### 6.1.3. Utilisation de requêtes paramétrées.

Le SQL permet de définir et d'enregistrer des requêtes sous un nom quelconque (exemple: "store xxx") sans en spécifier certains éléments ("placeholders"); à chacun de ceux-ci, l'utilisateur attribue une valeur lors de l'exécution de la requête (exemple: "start xxx (param. 1, param. 2)").

Ce mécanisme nous permet d'introduire plus facilement des OBJETS ou RELATIONS de chacun des types définis; chaque requete possède les paramètres correspondants; son exécution les introduit dans les colonnes adéquates et ajoute dirctement le type. Toute suppression est réalisée par un autre ensemble de requêtes, l'utilisateur spécifiant le nom de l'OBJET pour supprimer un OBJET d'un type particulier ou les noms de ceux-ci pour supprimer une RELATION particulière.

Bien que cet ensemble d'outils possède une certaine souplesse, il ne permet pas la transparence complète de la structure d'implémentation que les vues permettent de simuler (cfr. chapitre 5). Nous lui préférons un des deux derniers mécanismes présentés ci-dessous.

### 6.1.4. Utilisation de routines paramétrées.

Cet outil est la généralisation du précédent: il permet non seulement de paramétrer des requêtes mais aussi de rassembler celles-ci dans une routine exécutée en une fois (exemple: "run YYY (param. 1, param. 2)"). A titre exemplatif, nous avons implémenté quelques routines d'insertion et de suppression d'OBJETS. Celles-ci introduisent ou suppriment non seulement l'OBJET dont on donne les propriétés mais également les RELATIONS correspondantes, ceci masquant complètement la technique d'implémentation. La pauvreté du mécanisme de transfert des paramètres ( par valeurs uniquement) ne nous autorise pas à effectuer les vérifications de cohérence liées à toute modification de la BD, celles-ci restant à la charge de l'utilisateur.

Remarque: les paramètres doivent être donnés dans l'ordre où ils sont spécifiés ici.

### 6.1.4.1. Introduction - suppression d'un T.E.

- INSTE: introduction d'un T.E. et de son lien STA paramètres: nom du SCHEMA, nom du T.E..

- DELTE: suppression d'un T.E. et de son lien STA paramètres: nom du SCHEMA, nom du T.E..

### 6.1.4.2. Introduction - suppression d'un attribut.

- INSATTR1: introduction des propriétés d'un attribut de niveau 1 et son lien avec un T.E.
- paramètres: nom du T.E., nom, type et classe de l'attribut.

- DELATTR1: suppression d'un attribut et de son lien avec un T.E.
- paramètres: nom du T.E., nom de l'attribut.

- INSATTR2: similaire à INSATTR1 pour l'attribut d'un TAS.

### 6.1.4.3. Introduction d'un TAS.

- INSTAS: introduction d'un TAS binaire ainsi que les caractéristiques des rôles joués par chaque T.E. y participant; remarquons que ceci suppose l'insertion de trois OBJETS : un TAS et deux ROLES et de quatre relations entre:

  - TAS et ROLE (T.E.1)
  - TAS et ROLE (T.E.2)
  - T.E.1 et ROLE (T.E.1)
  - TE2 et ROLE (T.E.2)

  (cfr. chapitre 5)

- paramètres: nom du TAS, nom du premier T.E., nom, rôle, connectivité du rôle joué par le premier T.E., nom du second T.E., nom rôle, connectivité du rôle joué par le second T.E..

#### 6.1.4.4. Introduction d'un TRT

- INSTRT1 : introduction des propriétés d'un traitement ainsi que des liens avec son créateur et le service pour lequel il est créé.

- paramétres : nom du SERV, nom de la PERS, nom, type, étape, date de dernière mise-à-jour, développement du traitement.

- INSTRT2 : similaire à INSTRT1 à l'exception du lien : TRT - SERV.

#### 6.1.4.5. Une routine plus complexe: INSTSQL.

##### 6.1.4.5.1. Présentation.

Dans le but de compléter nos exemples, nous avons implémenté une routine réalisant la plus grande partie de la correspondance entre la description d'une table SQL par le DD et celle qui est enregistrée dans les tables systèmes du SGBD lui-même.
Cette routine permet d'introduire en une fois:

- un SCHEMA rassemblant la description de cette table et le lien avec son créateur .

- un T.E. correspondant à la table ainsi que son lien avec le SCHEMA.

- un FICHIER correspondant à la table ainsi que son lien avec le T.E..

- l'ensemble des attributs simples, élémentaires et obligatoires ou facultatifs (si une valeur nulle est autorisée) correspondant à chaque colonne ainsi que leurs liens avec le T.E..

Cette routine ne traduit ni les "unique index" éventuels (IKO), ni les privilèges pour d'autres utilisateurs, ni évidemment les correspondances possibles entre les éléments de ce SCHEMA et d'autres.
Pour plus de facilité, nous unifions le nom de la table et les noms du SCHEMA, du FICHIER, du T.E. et du "ROLESP".

Ajoutons pour terminer que la routine crée deux tables intermédiaires OBJ2 et REL2 pour permettre l'introduction en deux fois du nom et du type de certains objets, ceux-ci étant obligatoires dans les tables de départ. OBJ2 et REL2 sont ensuite supprimés.

- paramètres: nom du créateur (pour le DD), nom du créateur (pour les tables système c'est-à-dire nom de la MV), nom de la table.

6.1.4.5.2. Problème sous-jacent.

Nous pouvons remarquer que cette routine présente une grave erreur quant à la correspondance de certains identifiants. En effet, chaque attribut décrit dans le DD (de même que chaque OBJET d'un type quelconque) est identifié uniquement par son nom. Ceci résulte d'un choix d'implémentation (cfr. chapitre 5). Par contre, le nom de la colonne ("cname") auquel il est unifié ici ne constitue pas un identifiant pour la colonne id. (colonne) = "cname", "tname", "creator" (cfr. chapitre 2).

D'une part, ceci met en évidence un manque de souplesse du SDD. Nous détaillerons ce point dans l'évaluation (chapitre 7). D'autre part, nous pourrions réaliser la concaténation des trois valeurs identifiant une colonne et l'unifier au nom de l'attribut. Une telle manipulation de chaines de caractères est impossible ici, ce qui constitue un second inconvénient limitant l'usage des routines SQL (le premier étant la pauvreté du mécanisme de transfert par paramètres).

Tout ceci explique la raison pour laquelle nous avons conservé cet exemple erronné. Un dernier outil plus puissant est la création de programmes d'applications par lesquels toute commande SQL peut être transmise et exécutée.

06/07/85 ROUTINES ENREGISTREES

```
NAME            SEQNO  COMMAND
--------  ------------  ------------------------------------------------------------------
DELATTR1            10  DELETE FROM OBJET WHERE TYPE_O='ATTR' AND NOM_O='&2'
                    20  DELETE FROM RELATION WHERE TYPE1_R='ATTR' AND NOM1_R='&2'-
                    30   TYPE2_R='TE' AND NOM2_R='&1'

DELTE               10  DELETE FROM OBJET WHERE TYPE_O='TE' AND NOM_O='&2'
                    20  DELETE FROM RELATION WHERE TYPE1_R='SCHEMA' AND NOM1_R='&1' AND-
                    30   TYPE2_R='TE' AND NOM2_R='&2'

INSATTR1            10  INSERT INTO OBJET VALUES ('ATTR','&2','&4','&3',NULL,NULL,1,NULL)
                    20  INSERT INTO RELATION VALUES ('ATTR','&2','TE','&1')

INSATTR2            10  INSERT INTO OBJET VALUES ('ATTR','&2','&4','&3',NULL,NULL,1,NULL)
                    20  INSERT INTO RELATION VALUES ('ATTR','&2','TAS','&1')

INSTAS              10  INPUT OBJET
                    20   'TAS','&1',NULL,NULL,NULL,NULL,NULL,NULL
                    30   'ROLE','&3',NULL,NULL,NULL,'&4',NULL,'&5'
                    40   'ROLE','&7',NULL,NULL,NULL,'&8',NULL,'&9'
                    50  END
                    60  INPUT RELATION
                    70   'ROLE','&3','TAS','&1'
                    80   'ROLE','&3','TE','&2'
                    90   'ROLE','&7','TAS','&1'
                   100   'ROLE','&7','TE','&6'
                   110  END

INSTE               10  INSERT INTO OBJET VALUES ('TE','&2',NULL,NULL,NULL,NULL,NULL,NULL)
                    20  INSERT INTO RELATION VALUES ('SCHEMA','&1','TE','&2')

INSTRT1             10  INPUT OBJET
                    20   'TRT','&3','&4','&5',&6,'&7',1,'N'
                    30   'ROLETP','&3',NULL,NULL,NULL,'CREAT',NULL,NULL
                    40  END
                    50  INPUT RELATION
                    60   'SERV','&1','TRT','&3'
                    70   'ROLETP','&3','TRT','&3'
                    80   'PERS','&2','ROLETP','&3'
                    90  END

INSTRT2             10  INPUT OBJET
                    20   'TRT','&2','&3','&4',&5,'&6',1,'N'
                    30   'ROLETP','&2',NULL,NULL,NULL,'CREAT',1,'N'
                    40  END
                    50  INPUT RELATION
                    60   'ROLETP','&2','TRT','&2'
                    70   'PERS','&1','TRT','&2'
                    80  END

INSTSJL             10  CREATE TABLE OBJ2 (TYPE_O2 VARCHAR(12),NOM_O2 VARCHAR(12),SSTYPE_O2 -
                    20  VARCHAR(12),SSCLASSE_O2 VARCHAR(12) ) IN STUDENT
                    30  INSERT INTO OBJ2 (NOM_O2,SSTYPE_O2,SSCLASSE_O2)-
                    40         SELECT CNAME,COLTYPE,NULLS-
                    50         FROM SYSTEM.SYSCOLUMNS-
                    60         WHERE TNAME='&3' AND CREATOR='&2'
                    70  UPDATE OBJ2 SET TYPE_O2='ATTR'
                    72  UPDATE OBJ2 SET SSCLASSE_O2='DES' WHERE SSCLASSE_O2='N'
                    74  UPDATE OBJ2 SET SSCLASSE_O2='FES' WHERE SSCLASSE_O2='Y'
                    80  INSERT INTO OBJET (TYPE_O,NOM_O,SSTYPE_O,SSCLASSE_O)-
```

08/07/85 ROUTINES ENREGISTREES

```
NAME              SEQNO  COMMAND
--------  ------------  --------------------------------------------------------------------------------
INSTSQL             90          SELECT TYPE_O2,NOM_O2,SSTYPE_O2,SSCLASSE_O2-
                   100          FROM OBJ2
                   110  DROP TABLE OBJ2
                   120  INPUT OBJET
                   130   'SCHEMA','&3','SE',NULL,NULL,'T',NULL,NULL
                   140   'TE','&3',NULL,NULL,NULL,NULL,NULL,NULL
                   150   'FICHIER','&3','TABLESQL','PERM',NULL,NULL,NULL,NULL
                   160   'ROLESP','&3',NULL,NULL,NULL,'CREAS',NULL,NULL
                   170  END
                   180  INPUT RELATION
                   190   'FICHIER','&3','TE','&3'
                   200   'SCHEMA','&3','TE','&3'
                   210   'ROLESP','&3','SCHEMA','&3'
                   220   'PERS','&1','ROLESP','&3'
                   230  END
                   240  CREATE TABLE REL2 (TYPE1_R2 VARCHAR(12),NOM1_R2 VARCHAR(12),TYPE2_R2 -
                   250  VARCHAR(12),NOM2_R2 VARCHAR(12) ) IN STUDENT
                   260  INSERT INTO REL2 (NOM1_R2) SELECT CNAME-
                   270                             FROM SYSTEM.SYSCOLUMNS-
                   280                             WHERE CREATOR='&2' AND TNAME='&3'
                   290  UPDATE REL2 SET TYPE1_R2='ATTR',TYPE2_R2='TE',NOM2_R2='&3'
                   300  INSERT INTO RELATION SELECT * FROM REL2
                   310  DROP TABLE REL2
```

### 6.1.5. Programme d'application.

#### 6.1.5.1. Données du problème.

Nous allons donner les spécifications des différents composants d'un petit progamme d'introduction d'éléments dans le dictionnaire de données. Nous nous limiterons au cas suivant. Etant donné un SCHEMA défini, introduisons les différents éléments de sa description : types d'entités, d'association limités aux T.A. binaires, attributs ainsi que les liens entre ces éléments entre eux et vis à vis du SCHEMA.(Nous ne considèrons ni l'IKO ni les FICHIERS.) Cette introduction respectera les principales contraintes associées et prévoira des recouvrements d'erreurs au clavier ou dues au systéme.

#### 6.1.5.2. Composants de base.

##### 6.1.5.2.1.Découpe générale

Dans une approche "top-down", distinguons les quatre composants généraux de notre programme correspondant à l'enregistrement d'informations des différents types considérés ainsi que les régles de précédence qu'ils doivent vérifier, pour conserver la BD cohérente.

intro-T.E. :

Introduction d'un T.E. et de son lien avec le SCHEMA.
Contrainte : le schéma est défini.

intro-TA-binaire

Introduction d'un TA binaire et des rôles joués par chaque TE.
Contrainte : les deux T.E. sont définis

intro-ATTR-N1
---

Introduction d'un attribut de niveau 1 c'est-à-dire non issu de la décomposition d'autres attributs, de ses différentes caractéristiques et de son lien avec le T.E. ou le TAS.
Contrainte : le T.E. ou le TAS est défini.

intro-ATTR-NI
---

Introduction d'un attribut de niveau i ($i \geq 1$) c'est-à-dire issu de la décomposition d'un autre attribut, de ses différentes caractéristiques et de ses liens avec le T.E. ou le TAS et l'attribut décomposable.

Contrainte: le T.E. ou le TAS et l'attribut décomposable sont définis.

### 6.1.5.2.2. La notion de transaction

Dans le but de conserver la BD cohèrente à tout moment, nous devons définir, pour chaque type d'information enregistrée, l'ensemble minimum d'opérations de mise-à-jour à effectuer sur la BD, soit complétement, soit pas du tout. Cet ensemble est traduit par une séquence atomique d'opérations c'est-à-dire une transaction. Nous verrons que dans notre cas, cette notion doit etre détaillée. Nous considérons que les niveaux inférieurs devront implémenter les deux primitives: "début d'introduction" et "fin d'introduction".

### 6.1.5.2.3. Composants d'une introduction.

Toute introduction de données va être réalisée par des primitives détaillées aux niveaux inférieurs. Celles-ci sont liées soit à la lecture c'est-à-dire la partie interactive organisant l'introduction des données par l'utilisateur, soit au SGBD c'est-à-dire d'une part les requêtes permettant de vérifier les différentes contraintes que nous avons détaillées et d'autre part l'enregistrement des informations dans la BD SQL.

Nous savons que ce niveau devra implémenter la notion de transaction, notamment par les deux primitives citées ci-dessus. Nous allons rapidement passer en revue les primitives de chaque composant et plus précisément celles du SGBD, la lecture ne posant pas de problèmes.

#### 6.1.5.3. La lecture

##### 6.1.5.3.1. Lire-nom-élément.

Introduction du nom identifiant d'un élément quelconque (SCHEMA, T.E., TAS, ATTR) soit pour enregistrer ce nouvel élément, soit pour vérifier une contrainte sur celui-ci. Dans ce cas, il faut prévoir en outre la possibilité d'abandon par l'utilisateur, si plusieurs tests infructueux ont été effectués.
Paramètres en entrée : type élément, but lecture
Paramétres en sortie : nom, abandon.

##### 6.1.5.3.2. Lire-propriétés-éléments.

Introduction des différents attributs d'un élément (limités dans notre cas à ceux du T.E.: ATTR et du T.A.: ROLE) pour les enregistrer.
Paramétres en entrée : type, nom élément
Paramétres en sortie : valeur des différents attributs (avec possibilité de valeur nulle)

#### 6.1.5.4. Le SGBD.

##### 6.1.5.4.1. Primitives.

Nous utilisons ici les termes OBJET et RELATION; en effet, ces notions liées à l'implémentation du DD sont générales et donc connues de tous les niveaux du programme.

exist OBJET
---

Vérification de la présence d'un OBJET particulier dans la BD.
Paramétres en entrée : type et nom de l'OBJET.
Paramétres en sortie : indicateur de présence ou d'absence.

stockage OBJET
---

Introduction d'un OBJET dans la BD, avec toutes les colonnes propres à chaque type (cas possibles ici : T.E.,TAS,ROLE , ATTR).
Paramétres en entrée :
- type et nom de l'objet
- propriétés propres à chaque type
  (avec possibilité de valeurs nulles).

stockage RELATION :
---

Introduction d'une RELATION dans la BD (cas possibles ici : STA, ROLE-TAS, ROLE-TE, LIEN-TE, LIEN-TAS, DECOMP).
Paramétres en entrée : type et nom du premier objet
type et nom du second objet
(suivant la convention définie au chapitre 5).

début d'introduction
---

Définition de l'origine d'une séquence logique d'instructions que le niveau supérieur assimile à une transaction.
Paramètre en sortie : accordé ou non

fin d'introduction
---

Définition de la terminaison d'une séquence logique d'instructions.
Paramètre en sortie : indicateur précisant que soit tous les stokages sont réalisés soit aucune modification n'est apportée à la BD depuis le début d'introduction (en cas de problème).

### 6.1.5.4.2. Problèmes d'implémentation en SQL

Nous avons défini la notion de transaction. Dans notre cas, l'introduction des données est interactive, ce qui nous impose une restriction comme nous allons le voir. En SQL, chaque transaction est définie en trois étapes :

- déclaration de début de transaction ("set autocommit off")

- suite des instructions SQL ("insert", "input", "update",...)

- déclaration de fin de transaction ("set autocommit on")

Avant toute commande ultérieure, il faut finalement préciser si l'ensemble des instructions doit être exécuté
- complétement ("commit work") ou
- pas du tout ("rollback work").

En outre, en cas de problème durant la transaction, un "rollback work" rétablit également la situation initiale.

La version actuelle du SGBD SQL interdit à tout autre utlisateur l'accès des pages mémoire (512 bytes) contenant tout article concerné par une instruction isolée ou par l'ensemble des instructions d'une transaction. Comme d'un autre côté, les données sont ici introduites interactivement, il est préférable, pour chaque transaction d'enregistrer séparément toutes les mises à jour durant la phase interactive et ensuite de lancer la transaction SQL. Ceci facilite et accélére le travail commun de plusieurs utilisateurs sur le dictionnaire de données, cas très courant en pratique.

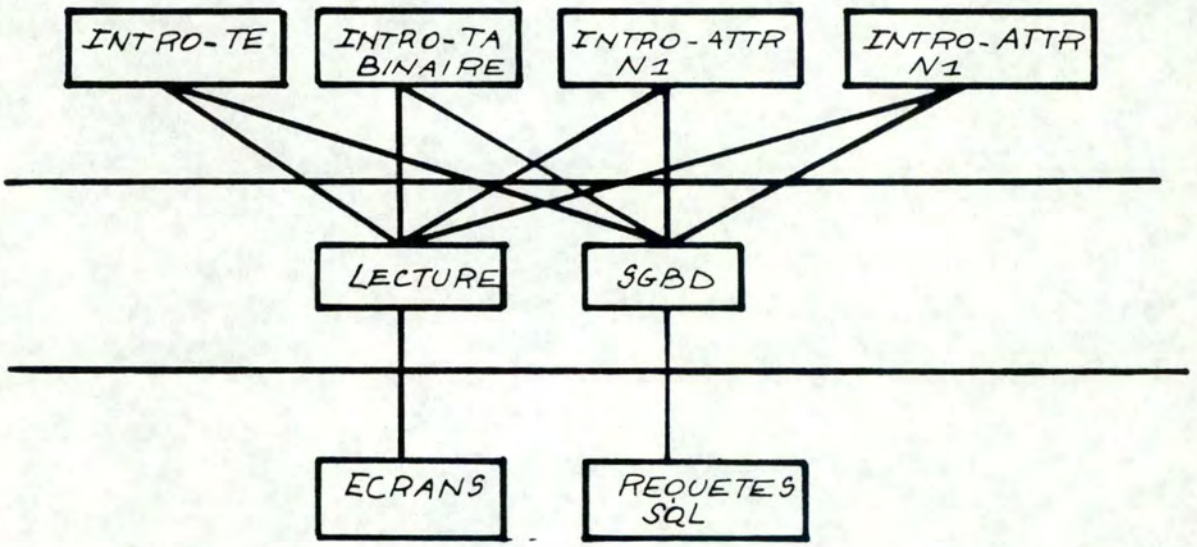
Examinons comment chaque primitive participe à l'implémentation de ces enregistrements en deux phases:

- le début d'introduction vide toute la zone mémoire réservée à l'enregistrement intermédiaire
- chaque stockage d'un objet ou d'une relation ajoute un enregistrement dans cette même zone intermédiaire
- la fin d'introduction met effectivement à jour les deux tables SQL sous forme de deux réelles transactions en transférant les enregistrements de la zone intermédiaire.

Une implémentation du stockage intermédiaire pourrait être sous forme de deux tableaux dont les colonnes sont la copie exacte des deux tables SQL.

6.1.5.5. Structure du programme
---

Nous obtenons la structuration des composants principaux du programme. Ceux-ci seront complétés par exemple par un module de sélection générale des types d'objets à introduire ou par un module d'introduction automatique de description de tables SQL via les tables système (cfr 6.1.4.5.)



## 6.2. PRODUCTION D'INFORMATIONS

### 6.2.1. L'outil le plus important : la manipulation des vues

Nous ne pouvons pas établir de parallèle entre les outils destinés à la production d'informations et ceux que nous avons développés pour la modification du contenu des tables. En effet, le mécanisme des vues ne nous a été pratiquement d'aucune utilité à l'étape précédente (cfr. 6.1.2). Par contre, un des buts de la technique d'implémentation est de permettre aux utilisateurs de produire eux-memes des informations à partir de celles-ci en créant leurs propres requetes c. à. d. en utilisant directement le langage du SGBD SQL qu'ils connaissent bien.

En conséquence, si nous pouvons faire la même restriction quant à l'utilisation directe des tables (cfr. 6.1.1), nous considérons la manipulation des vues comme l'outil principalement utilisé à ce niveau. Ce principe est d'application pour la manipulation directe, l'enregistrement de requêtes ou de routines paramétrées ou la création de petits programmes. Ce dernier mécanisme semble également moins utile et plus long à concevoir au vu des possibilités des autres outils. Nous allons reprendre quelques exemples de requetes simples et significatives sur le type de rapport que l'on peut obtenir directement.

### 6.2.2. Exemple de requêtes sur les vues

#### 6.2.2.1. Quelles sont les opérations du sous-module "XXX" pour lesquelles la conception physique est entammée? Parmi celles-ci, quelles sont celles qui sont prètes à etre encodées?

Nous supposons ici que la terminologie utilisée est celle que l'UCM a définie pour la conception d'une architecture physique (cfr 4.2.4).

Nous supposons également qu'une seule version est conservée pour chaque traitement (4.2.2.1).

Le rapport reprendra :

- le nom de l'opération (pour répondre à la première question)
- la date de dernière mise à jour (à titre informatif)
- le stade de développement (pour répondre à la seconde question)

```
select NOM-T, DMAJ-T, DEV-T
from   TRT
where  TYPE-T = 'OPERATION' and
       ETAPE-T = 'ARCHI-PHYS'
and    NOM-T in (select NOM-TFH
                 from   HIER
                 where  NOM-TPH in
                        (select NOM-T
                         from   TRT
                         where  NOM-T = 'XXX'
                         and    TYPE-T='SOUS-MODULE'))
```

Remarque : la question posée spécifie à la fois le nom, le type et l'étape du traitement; ces deux dernières propriétés sont redondantes, le nom étant un identifiant. Nous les avons cependant conservés dans cette première requête car ils n'introduisent pas d'erreur.

### 6.2.2.2. Quelles sont les différentes personnes qui développent les opérations du sous-module "XXX" ?

```
select NOM-P
from   PERS
where  NOM-P in (select NOM-PRTP
                 from   ROLETP
                 where  NOM-TRTP in
                       (select NOM-T
                        from   TRT
                        where  TYPE-T='OPERATION'
                        and    NOM-T in
                              (select NOM-TFH
                               from   HIER
                               where NOM-TPH = 'XXX')))
```

6.2.2.3. Quelles sont les structures de données sur
lesquelles travaille le traitement "XXX" ?
De quelle manière interviennent-elles ?

Le rapport comprendra :

- le nom du schéma (pour répondre à la première question)
- le rôle de cet "input-output" (pour répondre à la seconde question).

```
select NOM-SCHIO, ROLE-IO
from   I-O
where  NOM-TIO = 'XXX'
```

6.2.2.4. Quels sont les fichiers se partageant la
structure de données 'XXX' ? Quels sont les
T.E. implémentés par chacun de ces fichiers ?

Le rapport comprendra :

- le nom du fichier (pour répondre à la première question)
- le nom du T.E. (pour répondre à la seconde question).

```
select          NOM-FF, NOM-TEF
from            FTE
where           NOM-TEF in (select NOM-TE
                            from   TE
                            where  NOM-SCHTE = 'XXX')
order by        NOM-FF, NOM-TEF (asc)
format group (NOM-FF)
```

### 6.2.2.5. Quels sont les T.E. et leurs attributs du premier niveau qui font partie de la structure de données 'XXX' ? Donner les caractéristiques de ces attributs.

Nous considérons ici qu'un attribut de premier niveau est non issu de la décomposition d'autres attributs.

Le rapport reprendra :

- le nom du T.E.
- le nom, la classe et le type de l'attribut.

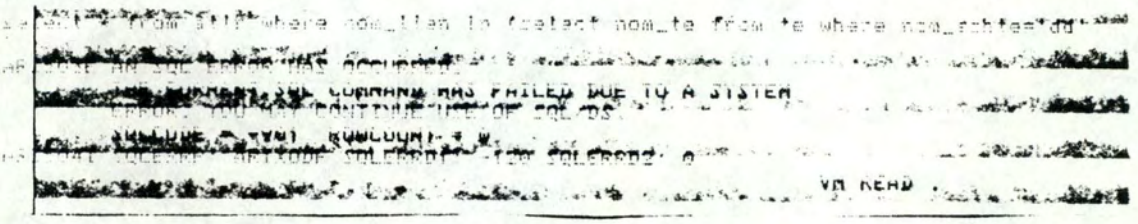
```
select          NOM-LIEN, NOM-A, CLASSE-A, TYPE-A
from            ATTR
where           NOM-LIEN in (select NOM-TE
                             from   TE
                             where  NOM-SCHTE = 'XXX')
order by        NOM-LIEN, NOM-A (asc)
format group (NOM-LIEN)
```

### 6.2.3. Problèmes d'implémentation

Au cours de nos différents tests d'implémentation de requêtes, nous avons eu de nombreux problèmes avec le SGBD SQL. Ceux-ci méritent d'etre mentionnés car ils ont sérieusement hypothéqué la dernière étape de notre travail : l'utilisation du dictionnaire de données (cfr. Chap.7).

Ces problèmes sont liés aux requêtes imbriquées sur des vues, cet outil étant la base de la production d'informations. Dans un premier temps, une partie de celles-là fonctionnait normalement alors que d'autres signalaient une grave erreur de l'utilisateur, bien que ces requêtes étaient similaires aux autres. Nos contacts avec le personnel de l'UCM nous ont convaincus que ce mauvais fonctionnement était du à une erreur dans un module du SQL.

Dans un second temps, une nouvelle version du SGBD a été implémentée à l'UCM (juillet 85). Nos derniers essais ont été systématiquement infructueux, le SGBD signalant une erreur système c'est-à-dire un problème non encore résolu dans le logiciel. Le message apparaissant systématiquement à l'écran est reproduit ci-dessous.



# CHAPITRE 7

# EVALUATION ET CONCLUSIONS

En vue d'évaluer l'adéquation du SDD aux besoins documentaires propres à l'UCM, nous examinerons successivement sa capacité documentaire et sa facilité d'exploitation.

Pour donner un point de départ à cette démarche, nous avions prévu d'introduire dans le DD la description de deux applications:

- le DD lui-même,

- le sous-module ASE c.à.d. l'ensemble des opérations interactives du module ASIGNAL (tenues des signalétiques) de l'application CAS à l'UCM.

Suite aux problèmes d'implémentation des outils de production d'information (cfr. 6.2.), l'objectif de démonstration de l'utilité d'un SDD à l'UCM par un test de fonctionnement pratique était partiellement compromis. En effet, les personnes concernées n'ont pu tester elles mêmes quelques requêtes et les confronter à leurs outils de documentation actuels.

Nous avons donc basé l'évaluation du SDD prototype sur des réflexions plus théoriques liées à la description de ces deux applications ainsi que sur les derniers entretiens que nous avons eus avec le personnel de l'UCM.

## 7.1. CAPACITE DOCUMENTAIRE

Le noyau du modèle tel que nous l'avons défini ne peut pas accueillir toutes les méta-données et les relations sur celles-ci décrivant complètement le SI de l'UCM. Cet inconvénient n'est pas d'une grande importance; en effet, nous avons écarté lors de la spécification et la conception certaines notions jugées non indispensables à l'organisation quitte à les incorporer ultérieurement via le mécanisme d'extensibilité si nécessaire. Nous en reprenons quelques exemples.

### 7.1.1. Adjonction de T.A. entre T.E. du même type

version schéma :

reliant les différentes versions d'un schéma pour un même niveau de développement (c'est-à-dire que les occurrences sont de même type.)

dépendance schéma :

reliant différents schémas de types différents, chacun décrivant une étape du cycle de vie du développement.

version traitement :

reliant les différentes versions d'un même traitement (exemple: mise-à-jour du code source, correction des spécifications,...)

dépendance traitement :

reliant par exemple les spécifications au code source d'un même programme.

Ces deux T.A. récursifs sur chaque T.E. précisent des relations plus fines; celles-ci sont justifiées pour certaines méthodes de développement et pas pour d'autres. Ceci est un exemple type d'extension particulière à une organisation.

déclenchement traitement :

reliant les divers composants d'un même niveau hiérarchique pour en préciser l'ordre et les conditions d'exécution.

Ce T.A. traduit un concept que nous avons inclus dans le T.A. HIER (4.2.3). En limitant la sémantique de ce dernier à la notion de hiérarchie, il est nécessaire d'ajouter un nouveau T.A. Aucun standard de méthodologie de développement de logiciel ne nous permet d'opter pour l'une ou l'autre de ces solutions. La seule description hiérarchique des applications de l'UCM nous a fait opter pour la première (cfr chapitre 2).

copie locf :

reliant chaque copie d'un fichier physique à un original unique.

version locf :
---

reliant les différentes versions d'un fichier dans l'ordre chronologique.

Ce niveau de détail ne semble pas requis au départ. Il est dépendant d'une terminologie (copie, version) spécifique et utile dans ce seul cas.

### 7.1.2. Adjonction de T.A. entre T.E. de types différents

privilèges accordés:
---

- par T.E. à PERS
- par ATTR à PERS

Ces T.A. permettent de traduire la sémantique de certaines notions reprises dans les tables système SQL. Nous avons préféré les T.A. ROLESP et ROLETP nécessitant dans certains cas (cfr.chapitre 4) l'introduction de nouveaux schémas. Nous avons supposés ces situations exceptionnelles et par définition non générales.

lien entre I-O et LOCF :
---

Nous avons traduit les notions de la statique des traitements (4.3.) en privilégiant l'ensemble des définitions d'une structure de données. Il est certain que ce choix suppose une lourde manipulation de SCHEMA si une description plus fine de la mémoire du SI est à représenter. Pour nous, ce type d'entité joue un rôle central dans la description, ce qui nous lie à plusieurs méthodologiques modernes de conception de BD.

L'adjonction d'un T.A. entre I-O et LOCF permet une fine description de la statique des traitements à l'étape d'implémentation. Bien qu'il introduit une redondance, nous préférons cette double solution à d'autres qui sont intermédiaires ou limitant l'importance de SCHEMA. Exemples :

- associer TRT à LOCF
- remplacer le T.A. I-O par un lien entre TRT et FICHIER.

machine virtuelle
---

cette notion propre à un O.S. de l'UCM (cfr chap.2) peut être représentée par :
- un T.A. entre RES et PERS
- un T.A. entre RES et SERV.

Ceci précise l'appartenance unique d'une machine virtuelle à un service et la possibilité pour plusieurs utilisateurs d'en utiliser plusieurs selon le service pour lequel ils travaillent à certains moments.

Cette notion particulière est mieux décrite par un T.E. (cfr ci dessous) du fait de l'éclatement possible du T.E. RES (cfr 4.4.)

### 7.1.3. Adjonction de T.E. et de T.A. associés

machine virtuelle :
---

cette notion est mieux représentée par un T.E.; on peut lui associer un nom, un rôle particulier dans l'organisation, ...Elle est reliée à PERS et à SERV. Plutôt que de l'associer à RES, elle sera directement utilisée pour un TRT ou un LOCT. Ceci traduit une première décomposition de RES. Il pourrait être utile d'en ajouter d'autres (ordinateur, imprimante, terminal, éditeur, compilateur, SGBD,...) Un manque d'information ne nous a pas permis de développer ces notions liées aux ressources. Il est vraisemblable que l'UCM n'en ait pas l'utilité actuellement.

DBSPACE :
---

cet autre concept est lié aux ressources. Il est défini comme partage de la mémoire auxiliaire allouée au SGBD entre les tables SQL. Nous ne l'avons pas représenté pour les mêmes raisons que précédemment. Ici, il sera relié à LOCF et (nous pouvons ajouter) à LOCT.

### 7.1.4 Adjonction d'entités de type MAPPING

Nous avons vu la signification générale de ce T.E. (chapitre 3) et sa méthode d'implémentation (chapitre 5) Citons quelques exemples d'utilisation.

une vue SQL:
---
peut être associée à la (aux) table(s) sur laquelle (lesquelles) elle est construite.

Pour cela, nous pouvons faire un MAPPING :
- entre : - un (des) FICHIER(S) et
  - un FICHIER
- entre : - un (des) T.E. et ses (leurs) ATTR et
  - un T.E. et ses ATTR
- entre des combinaisons de ceux-ci

la description d'une structure :
---
plutôt qu'un T.A. récursif précisant la dépendance de différents SCHEMAS, un MAPPING peut relier plus précisément toute description d'une sous-structure à son correspondant (T.E., TAS; ATTR, IKO, FICHIER).

## 7.2. FACILITE D'EXPLOITATION

### 7.2.1. Contraintes propres à l'implémentation du modèle

L'implémentation présente les nombreux avantages de l'extensibilité et ceux liés aux T.E. TEXTE et MAPPING. Nous en avons repris quelques uns. L'inconvénient principal est l'obligation d'identifier tout T.E. ou tout T.A. avec attributs par un seul attribut propre (cfr chapitre 5). Certains de ceux-ci sont non significatifs mais peuvent être standardisés :

par copie :
---
- nom du ROLETP = nom du TRT si ce lien en précise le créateur.
- nom du ROLESP = nom du SCHEMA si ce lien en précise la création.

par concaténation:
---
c'est une solution valable pour la plupart des autres T.A. avec attributs : LOCT, ROLETP et ROLESP (dans le cas de privilèges), I-O, ROLE, COMP1 et COMP2.

Cette standardisation ne pose pas de problèmes en cas d'introduction assistée des données; elle est par contre très lourde pour des insertions manuelles. La technique d'implémentation elle-même défavorise très largement l'insertion manuelle (cfr chapitre 6 : manipulation directe des tables).

Il serait possible d'identifier chaque composant par deux attributs : un premier qui lui est propre et un second dépendant d'un autre composant auquel celui-ci est relié. Cependant, cette caractéristique devrait être appliquée à tous les objets du modèle, ce qui, dans certains cas, est peu intéressant (SERVICE, RES,...) et entraine une lourdeur de manipulation (une colonne en plus dans la table OBJET).

Une autre technique d'extensibilité est d'implémenter directement les différents éléments du noyau sous forme de tables SQL. Une table supplémentaire OBJET serait la structure d'accueil unique pour toute occcurrence d'un nouveau type . Elle serait nécessairement reliée à toutes les autres par la création d'une nouvelle table, RELATION, jouant le même rôle que celle que nous avons implémentée, mais limité à ces objets particuliers. (Cette technique est la transcription dans un SGBD relationnel du principe d'extensibilité développé par le SDD: IDD/CULLINET)

Si cette technique a des avantages incontestables : manipulation directe de tables, souplesse de définitions d'identifiants, elle considère l'extensibilité comme exceptionnelle et lourde à utiliser. Celle que nous avons choisi est très souple pour toute adjonction (ou même modification et suppression) aux types prédéfinis dans le noyau. Elle est ainsi très intéressante pour une BD susceptible d'être fortement modifiée comme c'est le cas dans notre environnement trop peu défini.

### 7.2.2. Outils de maintenance

Quelques exemples ont été implémentés. Ceux-ci nous ont également servi pour l'introduction des deux applications-test. Bien qu'ils soient élémentaires (quelques routines paramétres), ils reflètent clairement les notions définies dans le DD et constituent ainsi un bon outil de démonstration. Ils sont simples à utiliser et permettent l'introduction rapide d'une description.

### 7.2.3. Outils de conversion

La routine INSTSQL ne nous a pas permis d'introduire plus d'une description de table (les suivantes violant la contrainte sur l'identifiant dans la table OBJET).

### 7.2.4. Outils d'interrogation

Nous avons signalé les différents problèmes de requêtes imbriqués sur des vues, mécanisme particulièrement utile à ce niveau (cfr 6.2.3). Nous n'avons donc pas pu implémenter les différentes requêtes décrites.

Le seul moyen de pallier cet inconvénient actuel du SGBD est d'effectuer nos requêtes directement sur les tables OBJET et RELATION. Cette technique est beaucoup plus complexe car elle nécessite de la part de l'utilisateur de connaître le mécanisme d'implémentation. Ceci est, par définition, opposé au but poursuivi.

Pour donner une idée plus précise du contenu des deux tables, l'annexe 7.1. en présente l'extension.

# ANNEXE 2.1.

## LE DOSSIER INFORMATIQUE DE LA CAS

CAS CAISSE WALLONNE D'ASSURANCES SOCIALES DES CLASSES MOYENNES

DOSSIER INFORMATIQUE CHAPITRE 1 - GENERALITES



*******

## A - PRESENTATION DU DOSSIER

- Quel est le but poursuivi ?
- Quelles sont les informations contenues dans le "dossier informatique" de la C.A.S.?
- Comment sont-elles structurées ?
- Comment accéder à ce dossier et aux différentes parties qui le composent ?
- Comment est-il tenu à jour ?
- Qu'en est-il des programmes ? Où se situent-ils dans le dossier ?

*******

### A1 - Le but

Compte tenu :
- de l'ampleur de l'application informatique,
- de la complexité croissante,
- de la volonté des dirigeants de ce service de recourir davantage à l'informatique,

il apparait dès lors impérieux de fournir à ces mêmes dirigeants un outil qui leur permettent de :
- maintenir un controle efficace et constant sur l'informatisation,
- donner à l'application elle-même le caractère de portabilité et de pérennité qu'il convient de lui donner.

*******

### A2 - Le contenu

Toutes les informations jugées nécessaires et suffisantes dans la composition d'un dossier informatique sont réparties en cinq chapitres, à savoir:

- Chapitre 1 - Les généralités
  - présentation du dossier,

- présentation de la CAISSE D'ASSURANCES SOCIALES - ( C.A.S.)
- description générale du traitement informatique:
  - organigramme général,
  - description générale des modules de traitement.

- Chapitre 2 - Les modules
  - description des fonctions assignées à chaque module,
  - décomposition du module en sous-modules.

- Chapitre 3 - Les sous-modules
  - description des fonctions assignées à chaque sous-module,
  - décomposition du sous-module en opérations.

- Chapitre 4 - Les opérations
  - description des fonctions assignées à chaque opération,
  - décomposition de l'opération en programmes,
  - description des fonctions de chaque programme.

5 - Chapitre 5 - Les fichiers

1 - l'inventaire des fichiers,
2 - la description du contenu de chaque fichier.

6 - Chapitre 6 - L'index

- il s'agit d'un index de composition de chaque module, sous-module et opération.

*******

## A3 - Le contenant

Les informations contenues dans ce dossier sont mémorisées dans un espace-disque propre a la CAS. Elles sont structurées dans un mode hiérarchique: l'ensemble des traitements informatiques de la CAS, de meme que la description qui en est faite dans ce dossier, sont subdivisés en :

- modules,
- sous-modules,
- opérations,
- programmes.

Suivant les modalités de fonctionnement requises par la CAS, un module est composé de sous-modules. Un sous-module regroupe un ensemble d'opérations sous la meme fréquence d'exploitation: les opérations journalières, hebdomadaires, etc... .
Une opération est généralement un ensemble de programmes pouvant etre exécutés sans interruption dans l'ordre exact de leur numérotation. Une opération peut, cependant, ne comporter qu'un seul programme.

Un programme est un ensemble d'instructions symboliques que l'on peut généralement classer en trois types:
- instructions d'input: les données disponibles;
- instructions d'exécution: les opérations à réaliser;
- instructions d'output: les résultats à obtenir.

La découpe d'une opération en programmes est généralement fonction de la diversité des informations qui sont manipulées: c'est à dire du nombre de fichiers.
Il est, par ailleurs, demandé aux informaticiens de respecter dans la mesure du possible le principe suivant: plutot deux petits programmes qu'un gros !...

L'application de ce principe influence favorablement les temps de développement et de test d'un projet; elle s'impose chaque fois que:
- le programme traite des petits volumes en input et en output;
- le programme est exploité peu fréquemment.

Chaque module porte un nom symbolique.
Chaque module, sous-module, opération ou programme porte un nom-code. Le premier caractère d'un nom symbolique et d'un nom-code est toujours la lettre A, de Assurances...!
Le nom symbolique d'un module est composé de sept caractères.
Le nom-code d'un module est composé de deux caractères.
Le nom-code d'un sous-module est composé de trois caractères.
Le nom-code d'une opération est composé de quatre caractères, si cette opération en regoupe plusieurs autres.
Le nom-code d'une opération est composé de cinq caractères.
Le nom-code d'un programme est composé de six caractères.

La séquence et la signification des caractères qui composent un nom-code est la suivante:

XXXXXX
IIIIII__: le numéro de séquence du programme dans l'opération,
IIIII___: la codification de l'opération,
IIII____: la codification d'une opération principale,
III_____: la codification du sous-module,
II______: la codification du module,
I_______: le code A, propre à la CAS;

- le numéro de séquence d'un programme est compris entre 0 et 9 et/ou A à Z;
- le code de l'opération se compose de deux positions proches du nom du fichier principal traité ou encore du nom donné au traitement.
- le code du sous-module exprime la périodicité:
  - J : batch journalier
  - H : " hebdomadaire
  - Q : " bimensuel
  - M : " mensuel
  - T : " trimestriel
  - S : " semestriel
  - A : " annuel
  - O : " occasionnel
  - P : " provisoire
  - E : transaction-écran
  - I : requête infocentre (SQL).
- le code du module est un caractère proche du nom symbolique du module: S de ASIGNAL, pour le module de traitement des signalétiques.

Ces différents objets représentent, en fait, les moyens élaborés et mis en oeuvre pour obtenir le traitement souhaité. La matière à traiter est supportée par les fichiers. Il en existe un inventaire; les informations contenues dans chaque fichier sont décrites dans les dessins de fichiers au chapitre 5 .

L'index est une table contenant le nom-code et la signification de tous les objets: modules, sous-modules, opérations et programmes; les noms-codes sont classés par ordre alphabétique.

*******

## A4 - L'accès au dossier

**********

Comme dit ci-avant, le dossier est mémorisé sur support magnéti-

que. Pour avoir accès au dossier il faut préalablement:
1 - connecter un terminal à ce support par:

LOGON DOSCAS
suivi du mot de passe

2 - connaitre et donner au système la formule suivante:

X A DOSSIER A

chaque mot étant bien séparé par un espace.

Grace à cette formule le premier accès est établi. L'utilisateur dispose donc, dans ce premier temps, de toutes les informations relatives aux généralités. L'action combinée des touches ALT et PF8 lui permet de progresser dans le texte; les touches ALT et PF7 opèrent le retour à l'écran précèdent; les touches ALT et PF3 terminent la consultation.

A la fin des généralités, l'utilisateur découvre l'énoncé des modules de traitement, chacun de ceux-ci étant accompagné de son nom symbolique et de son nom-code.

Disposant du nom-code d'un module ( par exemple: AS de ASIGNAL, Gestion des Signalétiques et Barèmes ), l'utilisateur peut accéder à la description de celui-ci par la formule:

X AS DOSSIER A

La description d'un module présente les sous-modules dont il est composé. Le sous-module qui traite les opérations journalières sur les signalétiques et barèmes est accessible par la formule:

X ASJ DOSSIER A

Au terme de la description d'un sous-module se trouve l'énoncé des opérations qui le composent. Une opération, ( par exemple ASABC la mise à jour Annuelle du Barème des Cotisations ), est accessible par la formule:

X ASABC DOSSIER A

La description d'une opération contient les différents programmes qui la composent.

L'inventaire des fichiers est accessible par la formule:

X A FICHIER A

On trouve, par exemple, dans cet inventaire le fichier AFFILI - signalétique affiliés.

La disposition ( dessin ) et la signification des informations contenues dans un enregistrement-AFFILI sont accessibles par la formule suivante:

X AFFILI FICHIER A

L'index des objets est obtenu par la formule:

X A INDEX A

Somme toute et pour autant que l'utilisateur connaisse le nom de l'objet à atteindre, il lui suffit de mémoriser les trois mots-cléss suivants:

DOSSIER: pour obtenir la description d'un module, d'un sous-module, ou d'une opération;

FICHIER: pour obtenir l'inventaire des fichiers et la description du contenu de chacun.
INDEX : pour obtenir la table complète de tous les objets classés dans l'ordre alphabétique du nom-code.

*******

A5 - La tenue du dossier
**********

Par instruction de la direction générale des services, la tenue du dossier informatique est placée sous le controle direct du responsable du CTI. Celui-ci veille donc essentiellement à ce que d'une part, la maintenance du dossier soit effectuée correctement à l'occasion de la réalisation de tout nouveau projet ou encore lorsqu'une modification a été apportée à un programme existant et, d'autre part, à ce que les techniques coutumières, devenues des standards, soient respectées.

Tout informaticien chargé de la réalisation d'un projet, - à quel que niveau que se situe ce projet: module, opération ou programme-, est tenu d'actualiser le dossier par les explications qu'il juge nécessaires et suffisantes pour permettre au lecteur, informaticien ou non, de comprendre facilement et rapidement les objectifs assignés à tel module, à telle opération ou encore à tel programme.

*******

A6 - Les programmes
**********

Tous les programmes-sources de la CAS sont mémorisés dans une librairie ( espace-disque ) réservée à cette application.
Chaque programme-source ( c.à.d. le langage symbolique COBOL ou RPG ) est reproduit sur listing. L'ensemble de ces listings de programme se trouve au CTI. Cet ensemble est complété d'éléments plus techniques tels que les EXEC, AMSERV, SORTCTL et PANEL.

Il va de soi qu'un listing de programme ne s'adresse pas directement au non-informaticien pour lequel langage et technique de programmation ne présentent que très peu de signification. On suppose, d'une manière aussi évidente, que le programmeur chargé de modifier un programme écrit par une autre personne maitrise le langage et la technique à un point tel qu'il peut arriver à découvrir le sens de chaque instruction.

Pour permettre une compréhension plus rapide d'un programme, de la part d'un informaticien qui n'est pas l'auteur de ce programme, il est néanmoins demandé à quiconque terminant la mise au point d'un nouveau programme ou la modification d'un programme existant de décrire brièvement dans le programme-meme l'objectif qu'il a assigné a ce programme ainsi que les particularités et "astuces" auxquellles il a eu recours dans la solution.

*******
**********
*******

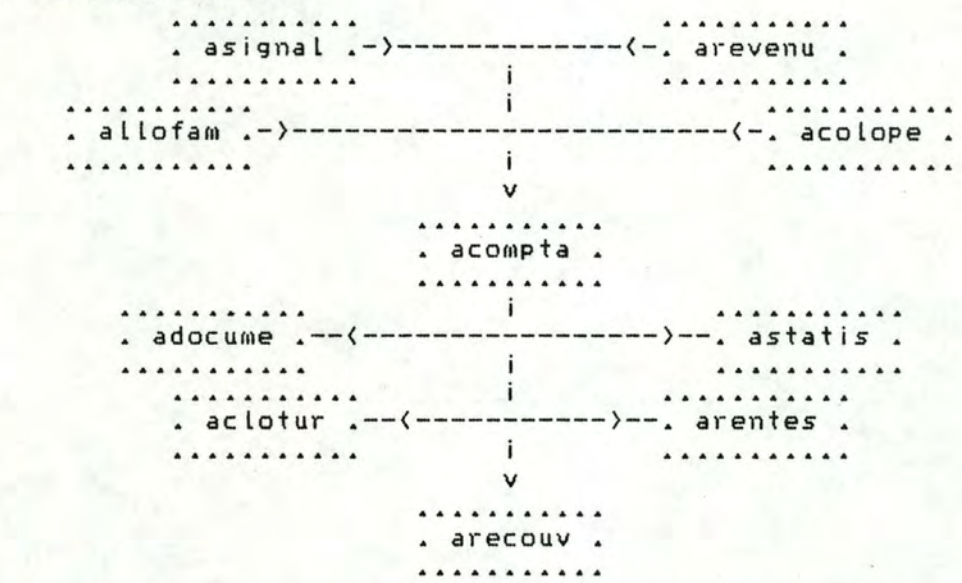
B - PRESENTATION DE LA CAISSE D'ASSURANCES SOCIALES
==========

tralisateurs: INASTI et INAMI;
- élaboration des clotures comptables;
- élaboration des statistiques;
- controle des comptes-affiliés;
- tenue de la comptabilité générale de la CAS.

*******
**********
*******

## C - DESCRIPTION GENERALE DU TRAITEMENT INFORMATIQUE.

### C1 - Organigramme général




1 - AS ASIGNAL: gestion des fichiers signalétiques et barèmes,

2 - AV AREVENU: traitement des revenus,

3 - AF ALLOFAM: traitement des allocations familiales,

4 - AL ACOLOPE: collecte journalière des opérations à comptabiliser,

5 - AC ACOMPTA: comptabilisation journalière des opérations,

6 - AD ADOCUME: édition hebdomadaire des documents,

7 - AT ASTATIS: statistiques,

8 - AU ACLOTUR: clotures comptables,

9 - AP ARENTES: traitement des rentes,

10 - AR ARECOUV: recouvrement des cotisations.

### C2 - Description générale des modules

Dans sa logique, l'organigramme général présente un point central: le module ACOMPTA. Il est pour les autres modules à la fois point

de convergence et point de départ. Les informations qu'il traite lui viennent principalement des modules ASIGNAL, AREVENU, ALLOFAM et ACOLOPE. Ce sont, par ailleurs, les informations qu'il produit qui sont exploitées par les modules ADOCUME, ACLOTUR, ASTATIS, ARENTES et ARECOUV.

Ces informations ont pour nom:
- le fichier des mouvements du trimestre,
- le fichier des primes attendues ou soldes.

Elles sont au coeur meme de la plupart des opérations comptables et administratives de la CAS.

1 - ASIGNAL

Ce module comporte toutes les opérations, en temps différe (BATCH) et en temps réel (T.P.), relatives a la tenue des informations signalétiques et des barèmes.

Plusieurs des informations contenues dans le signalétique affiliés et dans les barèmes sont consultées par le module ACOMPTA au moment du controle de validité et du traitement des mouvements comptables.

2 - AREVENU

Pour une année donnée, située dans la période d'assujettissement d'un affilié, les revenus professionnels de cette année représentent l'origine et la base essentielle du traitement de ce dossier.

Lorsque la CAS recoit un revenu que lui transmet l'INASTI le module AREVENU:
- prend en charge ce revenu;
- détermine la période couverte par ce revenu
  - soit une période de début d'activité,
  - soit une période d'activité définitive,
  - soit les deux;
- reconstitue l'historique des mouvements de la période couverte;
- détermine l'opération à effectuer:
  - soit un enrolement, si l'historique des mouvements du dossier ne contient aucun mouvement d'enrolement,
  - soit une régularisation, dans le cas contraire;
- effectue soit l'enrolement des cotisations pour la période couverte par ce revenu,
  soit une régularisation
  pour la période début d'activité,
  pour une période postèrieure a celle-ci,
  pour la période début d'activité et l'année qui suit immédiatement cette période.

Les mouvements d'enrolement et de régularisation générés par les opérations du module AREVENU sont ensuite traités par le module ACOMPTA.

3 - ALLOFAM

Sur la base du fichier signalétique enfants, le module ALLOFAM calcule au début de chaque mois le montant mensuel de l'allocation payable à chaque famille.

Le résultat du calcul est soumis à l'examen du service qui décide d'attribuer le droit ou de ne pas l'attribuer en fonction des éléments de controle qu'il détient.

Le module génère ensuite pour tous les droits attribués les mouvements-allocations qui seront pris en charge et comptabilisés par le module ACOMPTA.

Le module ALLOFAM procède encore, aux échéances déterminées, à l'édition des questionnaires, des certificats scolaires et des

CAS DOSSIER INFORMATIQUE CHAPITRE 2 - LES MODULES DE TRAITEMENT

## Module AS - ASIGNAL: GESTION DES FICHIERS SIGNALETIQUES ET BAREMES.

### 1. Sous-module ASJ - Les opérations BATCH journalières:

- Les objectifs:
  - collecte des mouvements
  - Création des nouveaux
  - Mutation
    - relatifs aux signalétiques: AFFILI - affiliés
      ATTRIB - attributaires
      ALLOCA - allocataires
      ENFANT - enfants.

### 2. Sous-module ASM - Les opérations BATCH mensuelles:

- Opération de tenue du fichier RGTI de l'INASTI ( Répertoire Général des Travailleurs Indépendants ).

- Opération de tenue du fichier UCMVIE de la B.I. ( Belgique Industrielle Cie d'Assurances ).

### 3. Sous-module ASO - Les opérations BATCH occasionnelles:

- Conversions diverses de fichiers: CMS, KSDS, SQL...

- Reconstitution après incident, ( RESTORE ).

### 8. Sous-module ASE - Les opérations T.P.:

- Les objectifs:
  - Consultation,
  - Création des nouveaux,
  - Mutation.
- relatifs aux signalétiques: AFFILI - affiliés
  ATTRIB - attributaires
  ALLOCA - allocataires
  ENFANT - enfants
  AFFPLC - affiliés PLC.

CAS DOSSIER INFORMATIQUE CHAPITRE 3 - LES SOUS-MODULES

Sous-module ASE - LES OPERATIONS T.P..

**********
A. * Signalétique affiliés - AFFILI *
**********

A1. Opération ASEFL - Consultation:

- Lecture de AFFILI.

A2. Opération ASEFC - Création:

- Tenue de AFFILI, AFFPLC, MVTPLC, UCMVIE, MVRGTI, AFFARC.

A3. Opération ASEFM - Mutation:

- Tenue de AFFILI, AFFPLC, MVTPLC, UCMVIE, PERDIP, MVRGTI, AFFARC.

++++++++++

**********
B. * Signalétique affiliés PLC - AFFPLC *
**********

B1. Opération ASEPL - Consultation de AFFPLC:

- Lecture de AFFPLC.

++++++++++

**********
C. * Signalétique attributaires - ATTRIB *
**********

C1. Opération ASETL - Consultation:

- Lecture de ATTRIB.

C2. Opération ASETC - Création:

- Tenue de ATTRIB, ALLOCA, ENFANT.

C3. Opération ASETM - Mutation:

- Tenue de ATTRIB, ALLOCA, ENFANT.

##########################################################################

**********
D. * Signalétique allocataires - ALLOCA *
**********

D1. Opération ASELL - Consultation:

- Lecture de ALLOCA.

D2. Opération ASELC - Création:

- Tenue de ALLOCA, ENFANT.

D3. Opération ASELM - Mutation:

- Tenue de ALLOCA, ENFANT.

##########################################################################

**********
E. * Signalétique enfants - ENFANT *
**********

E1. Opération ASEEL - Consultation:

- Lecture de ENFANT.

E2. Opération ASEEC - Création:

- Tenue de ENFANT.

E3. Opération ASEEM - Mutation:

- Tenue de ENFANT.

CAS DOSSIER INFORMATIQUE CHAPITRE 4 - LES OPERATIONS.

Opération ASEFC - Création des nouveaux signalétiques AFFILIES.

**********
A. * Les supports *
**********

A1. Les fichiers:

AFFILI, COMMUN, MVRGTI, AFFPLC, MVTPLC, UCMVIE, AFFARC.

A2. Les écrans:

- L'écran ASEFC1: les zones sont préfixées par E1-

```
;##############################################################################;

   CAISSE D'ASSURANCES SOCIALES                    SIGNALETIQUE-AFFILIES
   ============================                    =====================

                                C R E A T I O N
                                ---------------

            NUMERO DE DOSSIER                        CODE INASTI

            ... .... ..... ..                          ... ....

##############################################################################
```

- L'écran ASEFC2: les zones sont préfixées et suffixées par : E2-

```
##############################################################################
                  SIGNALETIQUE AFFILIES        CREATION
                  =====================        ========

NUM  ... .... ..... ..   DIG  .  CRE  .....  JNA  ..  NNA  ...........
ABF  ..  NOF ..................          PRF  .........
ABL  ..  NOL ........................    CJT  ...................
ADR  ..........................  BTE  ...     INA  ........
LOC  ................            BUR  .       INS  .....     CON  ....
CFA  ............                CFC  ............           PRO  .....
ENR  .  REV  ........  ANR  ..     ACR  ..
CODES:  CAT  ..    SEX   .    ECI   .    QUA   .    RLI   .    NAT ...  ..
======  MUC   .    PAN   .    PLC   .    PPE   .    GEN   .    RAF   .
        CST   .    AFF   .    NAC   ..   CMP   .    SRP   .    RRP   .
        DIP   :

TEA   ...  CEA  .  CAO  ...  TSA  ...  CSA  .  CAD  ...                  .
DEA   ......  DSA  ......  DCR  ......  DMJ  ......  DRA  ......

PLC:    DNC  ......  EFC  ......  FIC  ......  TYC  .  TCO  ...
====    PRP  .....  EXL  .  ENP  .  CEP  .  CSP  .  COP .

MESSAGE:
```

**********

**********
B. * Principes de fonctionnement. *
**********

- Le dialogue et les opérations se déroulent comme suit:

  - appel de la transaction ( programme ) ASEFC1;

  - affichage de l'écran ASEFC1;

  - introduction: - du numéro de dossier: E1-NUM,
    - du code INASTI: E1-INA;

  - controle: absence de numéro correspondant; voir C1

  - consultation du fichier des COMMUNES ( COMMUN ): voir C2
    pour la recherche sur base du code INASTI de la localité, du code INS, du code bureau des contributions, du régime linguistique;

  - définition du code bureau régional: BUR; voir C3

  - affichage de l'écran ASEFC2 comprenant les informations LOC, INS, CON, RLI, BUR;

  - modification éventuelle par l'utilisateur des informations LOC, INS, CON, RLI, BUR;

  - introduction des informations signalétiques en double brillance;

  - transfert dans des zones suffixées par N (...N) des informations contenues dans les zones E2-...

  - controles de validité; voir C4

  - nouvel affichage - les informations correctes:brillance normale,
    - les informations erronées: brillance double;

  - apres correction, génération de : AFFILI voir C5
    MVRGTI voir C6
    AFFPLC voir C7
    MVTPLC voir C8
    UCMVIE voir C9
    AFFARC voir C10

**********
C. * Détails des opérations. *
**********

C1. Controle: absence de numéro:

Signaler l'erreur si le numéro introduit existe dans AFFILI.

C2. Consultation du fichier des COMMUNES:

Si E1-INA1 ≠ 0
Recherche sur base du code INASTI ( E1-INA ) de:
- LOC: nom de la localité ( = CBE commune absorbée ),
- INS: code commune INS,
- CON: code du bureau de contributions,
- RLI: régime linguistique.

- Les lignes COMMUN marquées d'un code P dans la colonne PEC ne peuvent etre prises en considération.
- Si le code INASTI donné n'a pas de correspondant dans COMMUN, le programme en informe l'utilisateur qui introduit, de sa propre initiative, les informations requises.
- Transfert des informations extraites du fichier COMMUN dans les zones E2-...

C3. Définition du code bureau régional:

Si E1-INA1 ≠ 0
  E2-BUR est défini comme suit:
    SI E1-INA1 = 5, 6, 8, 9 :  E2-BUR = E1-INA1.
    Sinon : E2-BUR = 9.

C4. Controles de validité:

Sur les informations contenues dans les zones E2-...

Voir routine ASYVAF.

C5. Génération de AFFILI:

Apres validation et sur base des informations contenues dans les zones ...N

Déterminer le phonème dans PHO par routine ASYPHO,
Former DMJ par date système,
Si DCR = b
  former DCR par date système,
Générer un AFFILI au moyen des zones ...N
  NUM, DIG, CRE, NNA, JNA, ABF, NOF, PRF, ABL, NOL, CJT, ADR,
  BTE, INA, LOC, BUR, INS, CON, CFA, PRO, ENR, REV, ANR,
  ACR, CAT, SEX, ECI, QUA, RLI, NAT, MUC, PAN, PLC, PPE, GEN,
  RAF, CST, AFF, NAC, DIP, CMP, SRP, RRP, TEA, CEA, CAO, TSA,
  CSA, CAD, DEA, DSA, DCR, DMJ, PHO.
  Note: ARC, SEA, SSA, CZR, ORR = b .

C6. Génération de MVRGTI:

Apres validation et sur base des informations contenues dans les zones ...N

Si NUM1 = b
  compacter NOF dans NOFC par routine ASYNOF,
  transférer NOFC ( 16 positions ) dans NOFC15 ( 15 positions ),
  compacter PRF dans PRFC par routine ASYPRF,
  Si DCR = b
    former DAIV et DAIX par routine ASYDAT avec date système,
  Si DCR ≠ b
    former DAIV et DAIX par routine ASYDAT avec DCR
  Si AFF = 1, 2, 3
    générer un MVRGTI code V

| | | | | | |
|---|---|---|---|---|---|
| CID | = V | INAV | = INA | NACV | = NAC |
| CAS | = 019 | PROV | = PRO | DEAAMV | = DEAAM |
| NUM2 | = NUM2 | QUAV | = QUA | AFFV | = AFF |
| CRE | = CRE | RLIV | = RLI | JNAV | = JNA |
| DIG | = DIG | NOFCV | = NOFC15 | SEXV | = SEX |
| DAIV | | PRFCV | = PRFC | NAT2V | = NAT2 |

    générer un MVRGTI code W

| | | | |
|---|---|---|---|
| CID | = W | ADR1W | = ADR1 |
| CAS | = 019 | POSW | = POS |
| NUM2 | = NUM2 | LOCW | = LOC |
| CRE | = CRE | BTEW | = BTE |
| DIG | = DIG | | |

```
Si AFF = 4
   générer un MVRGTI code X
      CID  = X        TECX  = 14
      CAS  = 019      NACX  = NAC
      NUM2 = NUM2     TEAX  = TEA
      CRE  = CRE      NOFCX = NOFC
      DIG  = DIG      PRFCX = PRFC
      DAIX            INAX  = INA

   Note: toutes les autres zones = b .
```

C7. Génération de AFFPLC:

Apres validation et sur base des informations contenues dans les zones ...N

```
Si PLC = P
   Si NUM123 absent dans AFFPLC
      générer un AFFPLC au moyen des zones ...N
         NUM123, ABF, NOF, PRF, ADR, BTE, POS, LOC, CAT, NUM2,
         JNA, SEX, ECI, QUA, RLI, TSA, CSP, TEA, CEP, BUR, DIG,
         CJT, DNC, EFC, TYC, TCO, FIC, PRO, PRP, EXL, ENP, REV,
         ANR, ACR.
         Note: REI, BPR, DAE = b .
```

C8. Génération de MVTPLC:

Apres validation et sur base des informations contenues dans les zones ...N

```
Si PLC = P
   Transférer la date-système dans DAP
   Si NUM123 absent dans AFFPLC
      transférer un code C dans CCM
      générer un MVTPLC au moyen des zones ...N
         NUM123, NOF, PRF, DAP, CCM, "création".
   Si NUM123 présent dans AFFPLC
      transférer un code R dans CCM
      générer un MVTPLC au moyen des zones ...N
         NUM123, NOF, PRF, DAP, CCM, " re-création".
```

C9. Génération de UCMVIE:

Apres validation et sur base des informations contenues dans les zones ...N

```
Si PLC = P
   Si NUM123 absent dans AFFPLC
      transférer "C" dans CID
      générer un UCMVIE au moyen des zones ...N
         NUM123, ABF, NOF, PRF, ADR, BTE, POS, LOC, CAT, NUM2,
         JNA, SEX, ECI, QUA, RLI, TSA, CSP, TEA, CEP, BUR, DIG,
         CJT, DNC, EFC, TYC, TCO, FIC, PRO, CID.
```

C10. Génération de AFFARC:

Après validation et sur base des informations contenues dans les zones ...N

```
   Former DAM et TIM à partir de la date et l'heure système

Générer un AFFARC au moyen des zones ...N

   DAM, TIM
   NUM, DIG, CRE, NNA, JNA, ABF, NOF, PRF, ABL, NOL, CJT, ADR,
```

BTE, INA, LOC, BUR, INS, CON, CFA, PRO, ENR, REV, ANR,
ACR, CAT, SEX, ECI, QUA, RLI, NAT, MUC, PAN, PLC, PPE, GEN,
RAF, CST, AFF, NAC, DIP, CMP, SRP, RRP, TEA, CEA, CAO, TSA,
CSA, CAD, DEA, DSA, DCR, DMJ, PHO.
Note: ARC, SEA, SSA, CZR, ORR = b .

```
FILE: ASEFC     EXEC      B1  CLASSES MOYENNES NAMUR

&CONTROL OFF
SET DOS OFF
FILEDEF ASZCAS DISK ASZCAS DATA C
FILEDEF ASZNAC DISK ASZNAC DATA C
FILEDEF ASZNAT DISK ASZNAT DATA C
FILEDEF ASZPRO DISK ASZPRO DATA C
ASEFC
&RET = &RETCODE
&IF &RET = -561 &GOTO -PASDB
&IF &RET = 0 &EXIT
&TYPE !!!!! ERREUR &EXEC RETURN CODE &RET !!!!!!!!
&TYPE !!!!! PREVENIR LE CTI. !!!!!!!!!!!
SLEEP 3 SEC
&EXIT
-PASDB
&BEGTYPE
------------------------------------------------------------------------

TRAVAUX BATCHS EN COURS DE TRAITEMENT, ACCES IMPOSSIBLE ACTUELLEMENT

------------------------------------------------------------------------
&END
```

AFFILI: DB.
Signalétique des affiliés de la C.A.S.

AFFILIC: CMS.
Signalétique des affiliés de la C.A.S., à usage spécifique.

AFFILIK: KSDS.
Signalétique des affiliés de la C.A.S., interface entre AFFILI DB et les programmes RPG-BATCH.

AFFARC: DB.
Archivage de tous les mouvements signalétiques-affiliés datés et minutés.

AFFPLC: DB.
Signalétique des affiliés de la C.A.S. ayant souscrit à la P.L.C.: Pension Libre Complémentaire.

AFFPLCC: CMS.
Signalétique des affiliés de la C.A.S. ayant souscrit à la P.L.C.,interface entre AFFPLC DB et les programmes RPG-BATCH.

ALLOCA: DB.
Signalétique des allocataires de la C.A.S.

ALLOCAC: CMS.
Signalétique des allocataires, à usage spécifique.

ALLARC: DB.
Archivage de tous les mouvements signalétiques-allocataires, datés et minutés.

ATTRIB: DB.
Signalétique des attributaires de la C.A.S.

ATTRIBC: CMS.
Signalétique des attributaires, à usage spécifique.

ATTARC: DB.
Archivage de tous les mouvements signalétiques-attributaires, datés et minutés.

BONAFF: CMS.
Mouvements signalétique-affiliés sous la forme du bon d'encodage.

BONAFFS: CMS.
Mouvements signalétique-affiliés sous la forme du bon d'encodage générés par l'opération ASAF01 du module ASTATIS

COMMUN : DB.
Signalétique des COMMUNES.
Les lignes "communes périmées" sont marquées d'un code P dans la colonne PEC; elles ne peuvent etre prises en considération dans aucun traitement.

COMMUNC: CMS.
Signalétique des COMMUNES; interface entre COMMUN DB et les les programmes RPG-BATCH.

ENFANT: DB.
Signalétique des enfants de la C.A.S.

ENFANTC: CMS.
Signalétique des enfants, à usage spécifique.

ENFARC: DB.
Archivage de tous les mouvements signalétiques-enfants, datés et minutés.

MOUAFFC: CMS.
Mouvements signalétique-affiliés de CREATION: les différentes lignes du bon d'encodage étant regroupées en une seule ligne.

MOUAFFM: CMS.
Mouvements signalétique-affiliés de MUTATION: les différentes lignes du bon d'encodage étant regroupées en une seule ligne.

MVRGTI: DB.
Mouvements signalétiques des affiliés de la C.A.S. destinés à la tenue du fichier " Bande Carrière " du R.G.T.I. à l' I.N.A.S.T.I.: Répertoire Général des Travailleurs Indépendants, Institut National d'Assurances Social des Travailleurs Indépendants.

MVRGTIC: CMS.
Mouvements signalétiques des affiliés de la C.A.S. destines à la tenue du fichier R.G.T.I. présentés dans le dessin requis par le R.G.T.I.

MVTPLC: DB.
Dossiers P.L.C. mouvementés; opère la liaison entre la cellule P.L.C. et celles qui ont été à la base d'un mouvement signalétique relatif à un affilié P.L.C.

PERDIP: DB.
Historique par affilié des périodes ayant fait l'objet d'une dispense, d'une irrécouvrabilité ou d'une prescription.

PERDIPC: CMS.
Historique par affilié des périodes ayant fait l'objet d'une dispense, d'une irrécouvrabilité ou d'une prescription; interface entre PERDIP DB et les programmes RPG-BATCH.

UCMVIE: DB.
Mouvements signalétiques relatifs aux affiliés P.L.C. destinés à la tenue du fichier UCMVIE de la B.I.: Belgique Industrielle Cie d'assurances - Liège.

UCMVIEC: CMS.
Mouvements signalétiques relatifs aux affiliés P.L.C. destinés à la tenue du fichier UCMVIE, presentés dans le dessin requis par la B.I.

AFFILI DB
==========

| | | | |
|---|---|---|---|
| 14 | NUM | Numéro de dossier | (INDEX UNIQUE) |
| | NUM123 | 1 - 12 | |
| | NUM1 | 1 - 3 | Numéro de regroupement |
| | NUM11 | 1 - 1 | |
| | NUM23 | 4 - 12 | |
| | NUM2 | 4 - 7 | Année et mois de naissance |
| | NUM21 | 4 - 5 | Année de naissance |
| | NUM22 | 6 - 7 | Mois de naissance |
| | NUM3 | 8 - 12 | Numéro équivalant au compte de retraite |
| | NUM4 | 13 - 14 | Sous-numéro de dossier |
| 1 | DIG | Check digit | |
| 5 | CRE | Compte de retraite ( 5 dernières positions ) | |
| 11 | NNA | Numéro national | |
| 2 | JNA | Jour de naissance | |
| 2 | ABF | Abréviation de l'affilié: MR, MD, ML. | |
| 16 | NOF | Nom de l'affilié | |
| | NOF1 | 1 - 15 | |
| | NOF2 | 16 - 16 | |
| 8 | PRF | Prénom de l'affilié | |
| 2 | ABL | Abréviation de l'allocataire: MR. MD. ML. | |
| 21 | NOL | Nom de l'allocataire | |
| 20 | CJT | Nom du (de la) conjoint (e) | |
| 22 | ADR | Rue et numéro | |
| | ADR1 | 1 - 21 | |
| | ADR2 | 22 - 22 | |
| 3 | BTE | Numéro de boite | |
| 7 | INA | Code INASTI | |
| | INA1 | 1 - 1 | |
| | INA2 | 2 - 3 | |
| | POS | 4 - 7 | |
| 15 | LOC | Localité | |
| 1 | BUR | Bureau régional | |
| 5 | INS | Code commune INS | |
| 4 | CON | Code bureau des contributions | |
| 12/0 | CFA | Compte financier allocations | |
| | CFA1 | 1 - 10 | |
| | CFA2 | 11 - 12 | |
| 5 | PRO | Code profession | |
| | PRO1 | 1 - 2 | |
| | PRO2 | 3 - 5 | |
| 1 | ENR | Code enrolement | |
| 7 | REV | Revenu | |
| 2 | ANR | Année du revenu | |
| | ANR1 | 1 - 1 | |
| | ANR2 | 2 - 2 | |
| 2 | ACR | Actualité du revenu | |
| | ACR1 | 1 - 1 | |
| | ACR2 | 2 - 2 | |
| 2 | CAT | Code catégorie | |
| | CAT1 | 1 - 1 | |
| | CAT2 | 2 - 2 | |
| 1 | SEX | Code sexe | |
| 1 | ECI | Code état civil | |
| 1 | QUA | Code qualité | |
| 1 | RLI | Code régime linguistique | |
| 3 | NAT | Code nationalité | |
| | NAT1 | 1 - 2 | |
| | NAT2 | 3 - 3 | |
| 1 | MUC | Code mutuelle MUCLAMNA | |
| 1 | PAN | Code pension anticipée | |
| 1 | PLC | Code pension libre complémentaire | |
| 1 | PPE | Code prime de premier établissement | |

| | | |
|---|---|---|
| 1 | GEN | Code genre allocations familiales |
| 1 | RAF | Code retenue allocations familiales |
| 1 | CST | Code cotisations spéciale temporaire |
| 1 | AFF | Code affiliation |
| 1 | NAC | Code nature de cotisant |
| 1 | DIF | Code référence au fichier PERDIF |
| 1 | CMF | Code de compens. alloc. fam. |
| 1 | SRF | No de série Reg. de population (A.F.) |
| 1 | RRF | C. de rappel |
| 3 | TEA | Trimestre d'entrée |
| | TEA1 | 1 - 2 |
| | TEA2 | 3 - 3 |
| 1 | CEA | Code entrée |
| 3 | CAO | Caisse d'origine |
| 3 | TSA | Trimestre de sortie |
| | TSA1 | 1 - 2 |
| | TSA2 | 3 - 3 |
| 1 | CSA | Code sortie |
| 3 | CAD | Caisse de départ |
| 6 | DEA | Date d'entrée ( AAMMJJ ) |
| | DEAA | 1 - 2 |
| | DEAM | 3 - 4 |
| | DEAJ | 5 - 6 |
| 6 | DSA | Date de sortie ( AAMMJJ ) |
| | DSAA | 1 - 2 |
| | DSAM | 3 - 4 |
| | DSAJ | 5 - 6 |
| 6 | DCR | Date de création ( AAMMJJ ) |
| | DCRA | 1 - 2 |
| | DCRM | 3 - 4 |
| | DCRJ | 5 - 6 |
| 6 | DMJ | Date de la dernière mise à jour ( AAMMJJ ) |
| | DMJA | 1 - 2 |
| | DMJM | 3 - 4 |
| | DMJJ | 5 - 6 |
| 6 | PHO | Phonème |
| 1 | ARC | Code archivage du dossier |
| 1 | SEA | Code statistique entrée |
| 1 | SSA | Code statistique sortie |
| 1 | CZR | Code rappel-plan d'apurement |
| 1 | ORR | Origine du revenu |

# ANNEXE 5.1

# UN SCHEMA GENERAL

## 5.1.1. INTRODUCTION

Le schéma décrit ici englobe des notions beaucoup plus générales que l'objet même de ce travail. Elles peuvent servir de base pour le développement de toute application. En effet les concepts d'objets, de relations et de propriétés sont le point de départ de tout formalisme de description.*

Le schéma est brièvement décrit ici à l'aide du modèle E.A. enrichi de la notion de surclasse. Un exemple d'utilisation termine cette description. Auparavant, la notion de surclasse est rapidement introduite avec sa traduction dans le modèle E.A.

*Ces notions font notamment partie de "l'atelier logiciel" I.D.A. (F. N. D. P.) en collaboration avec le projet ISDOS (Université du Michigan).

## 5.1.2. NOTION DE SURCLASSE

### 5.1.2.1. Exemple

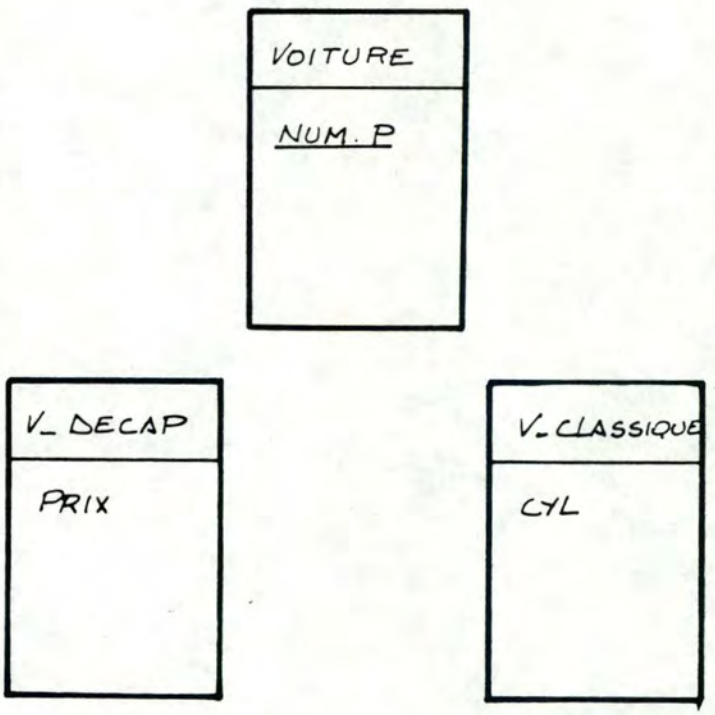
Supposons que le T.E. : VOITURE représente toutes les voitures circulant en Belgique et possédant une plaque d'immatriculation belge (en considérant que le terme "voiture" est sans ambiguité par rapport à "camion", "camionnette", "moto", "velo",..). Pour les différencier, nous identifierons chaque voiture par son numéro de plaque : NUM-P, un attribut de T.E.

Parmi ces voitures, nous désirons distinguer les voitures décapotables des autres que nous appellerons "classiques".

D'une part les voitures décapotables constituent un T.E. distinct car elles sont décrites par une ou des

propriétés qui leurs sont propres. Dans notre exemple elles sont caractérisées par leur PRIX (prix d'achat). Cet attribut a donc une valeur pour les seules voitures décapotables. Parallèlement, les voitures classiques constituent aussi un T.E. distinct. Elles et elles seules seront caractérisées par la CYL (cylindrée).

D'autre part, une voiture est un objet qui, en soi, est unique. Supposons que la voiture de NUM-P: ABC123 est décapotable, nous préciserons pour cette même voiture, son prix, par exemple 500.000 FRS.




En définitive, en considérant les occurrences, notre schéma est assez particulier : une même voiture fait partie de deux T.E. : VOITURE et VOITURE DECAPOTABLE ou VOITURE et VOITURE CLASSIQUE. Cette conclusion vient du fait que le T.E. voiture traduit un concept plus général que les deux autres T.E.

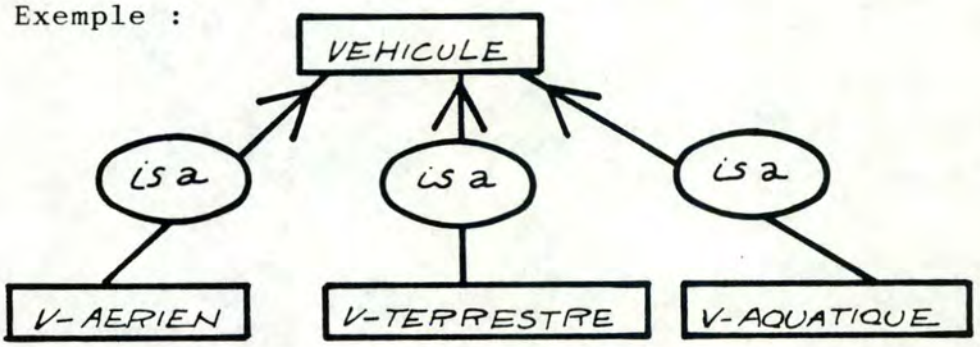
### 5.1.2.2. Principe

Généralisons notre exemple : le raisonnement suit le principe d'abstraction par généralisation / spécialisation, c'est-à-dire qu'il considére qu'une même entité peut appartenir à plusieurs types. Les types introduits définissent alors des classes, dont certaines sont plus générales et d'autres plus spécialisées, et dont chacune posséde ses propres attributs. Les entités qui font partie de plusieurs classe sont alors caractérisées par les attributs de chaque classe.

Cette notion de classe peut être affinée si on considère les deux types de découpes en niveaux.

### 5.1.2.3. Arborescence ou partition

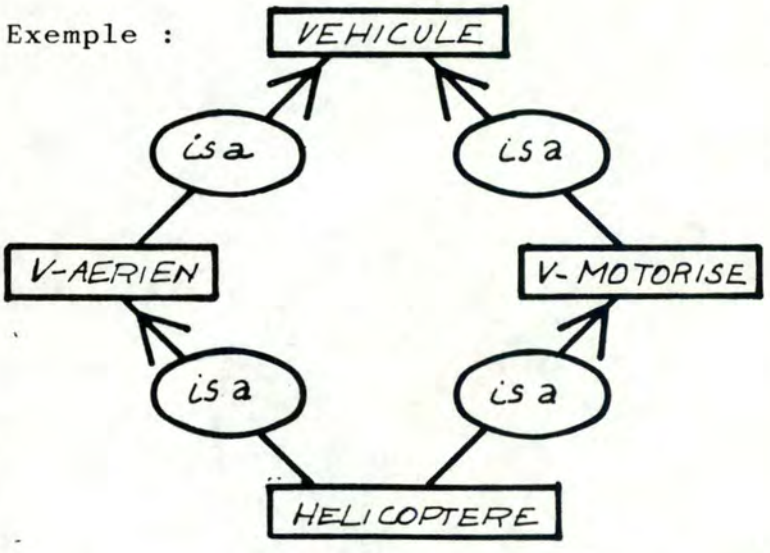
Exemple :




L'exemple montre une arborescence car les trois classes se recouvrent, une entité véhicule fait en plus partie d'une et d'une seule des trois sous-classe.

### 5.1.2.4. Hiérarchie

Exemple :



L'exemple montre ici des niveaux de spécialisations sans aucune contrainte sur l'appartenance d'une entité à une sous-classe. La seule limite est de conserver une hiérarchie c'est-à-dire que le graphe constitué par les classes et les associations "is a " ne doit pas contenir de circuits.

### 5.1.3. Expression de l'abstraction dans le modèle E.A.

Le schéma général décrit utilise la notion d'abstraction par généralisation / spécialisation. Par ailleurs, le modèle E.A. ne supporte pas cette notion. Il peut donc être bon de la traduire dans le modèle E.A. classique.

Nous nous limiterons à la traduction d'une arborescence car le cas d'une hiérarchie ne nous est pas utile ici.

Une arborescence peut être traduite par :
- nivellement par le bas
- nivellement par le haut
- introduction d'associations de connectivité: (1-1 et 0-1)

Considérons ces trois méthodes sur base d'un même exemple.

#### 5.1.3.1. Exemple (figure à la page suivante)

(cadre = personnel d'une compagnie d'aviation)

NB : AT, AH, AP représentent tous les attributs de chaque T.E.

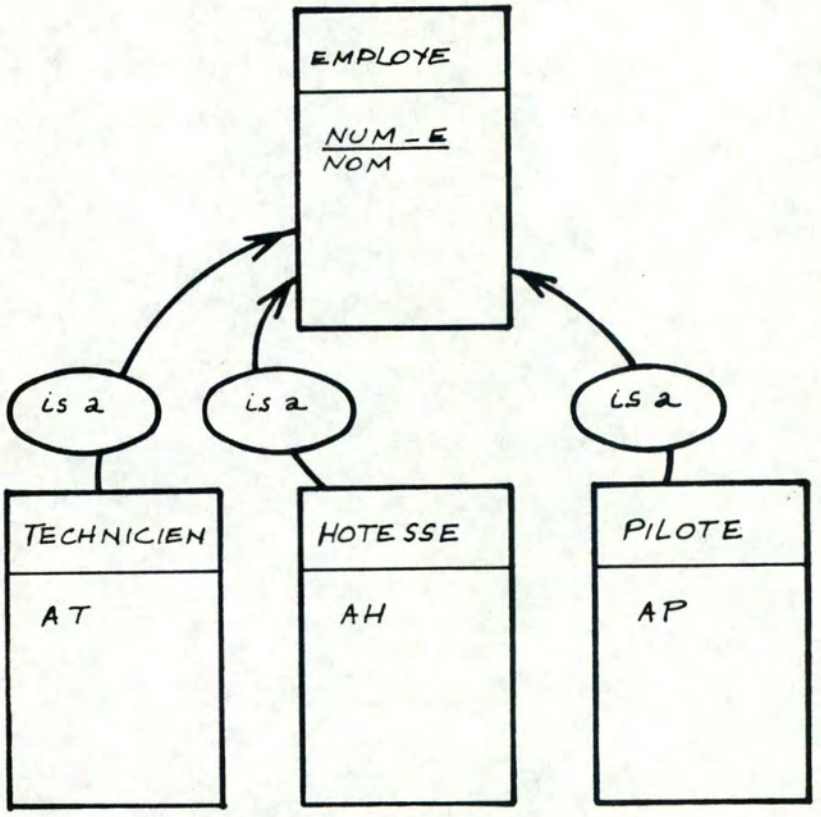
#### 5.1.3.2. Nivellement par le bas (figure page suivante)

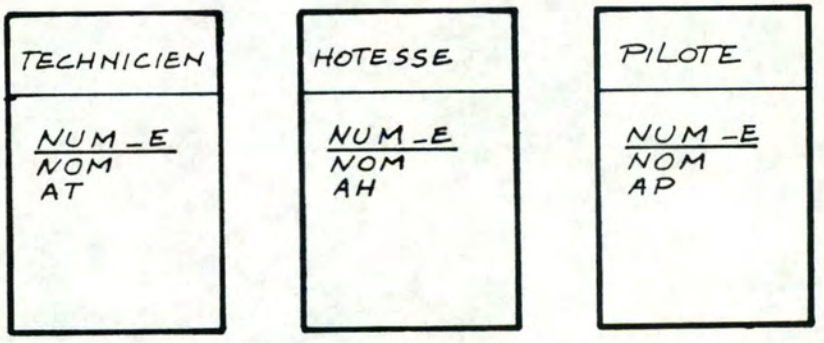
Observations :

La notion de surclasse dans laquelle faisaient

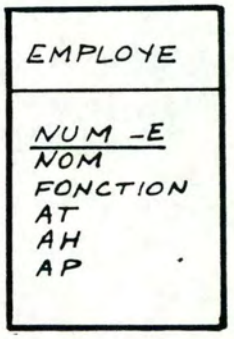
partie NUM-E et NOM est perdue. NUM-E est identifiant de chaque T.E. mais aussi de chaque employé, ce qui peut être traduit par une contrainte.

Tout T.A. éventuellement relié à EMPLOYE doit être remplacé par trois T.A. reliant chaque T.E.



### 5.1.3.3. Nivellement par le haut




Observations :

Cette représentation est plus abstraite tout en conservant chaque notion. Plus compacte, elle peut paraitre moins claire.

Les trois attributs AT, AH et AP sont facultatifs avec une contrainte d'exclusivité suivant la valeur de FONCTION.

Tout T.A. portant sur un des trois T.E. de départ est remplacé par un T.A. sur EMPLOYE en ajoutant une contrainte de connectivité suivant la valeur de FONCTION.

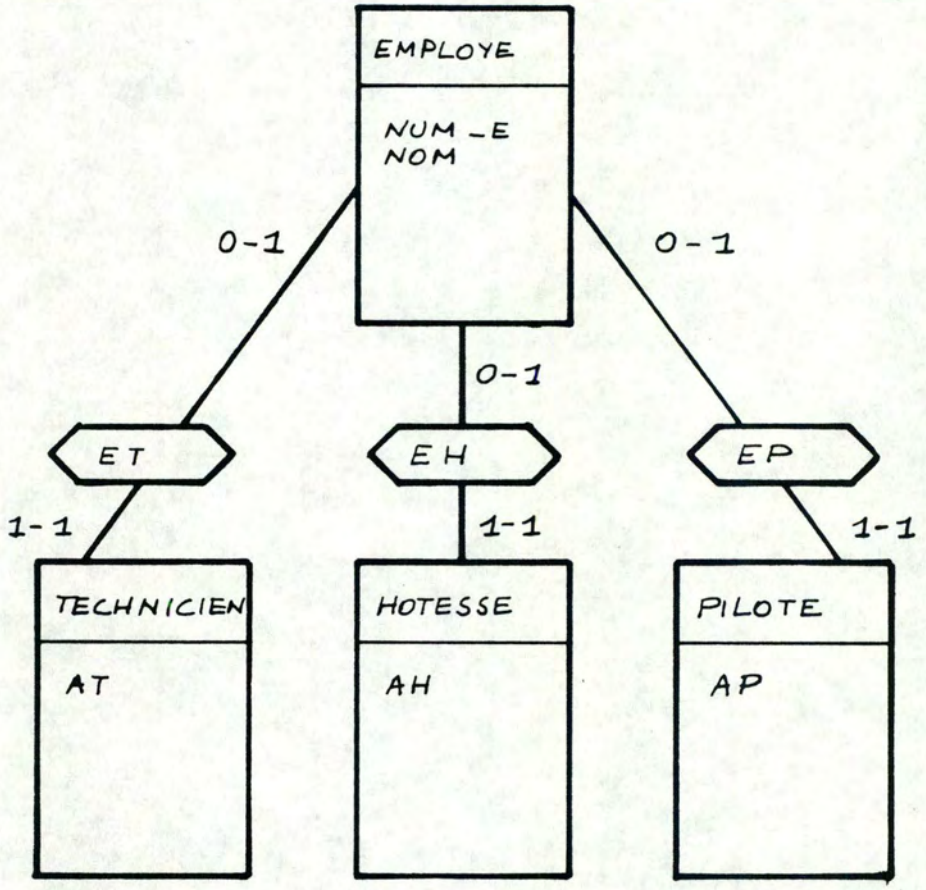
### 5.1.3.4. Introduction d'associations

(voir figure page suivante)

Observations :

Tous les T.E. de départ sont conservés. Un même objet est décrit à chaque fois par deux occurrences du modèle, ce qui est peu conventionnel.

Une contrainte d'exclusivité est ajoutée sur les trois T.A.: ET,EH et EP.


EMPLOYE
NUM_E
NOM
0-1
0-1
0-1
ET
EH
EP
1-1
1-1
1-1
TECHNICIEN
AT
HOTESSE
AH
PILOTE
AP

### 5.1.4 DESCRIPTION DU SCHEMA

Les cinq composants du schéma sont :

- élément
- objet
- relation
- propriété

} principaux

- texte } secondaire

Avant de les définir, nous allons distinguer les niveaux de description en présence.

#### 5.1.4.1. Niveaux de la description

Il est important de différencier la description du schéma de l'usage qui sera fait de ce schéma qui, dans ce cas-ci, est également descriptif.

D'une part, le schéma est décrit à l'aide du modèle E.A. qui est une méthode formelle de description d'éléments du réel perçu.

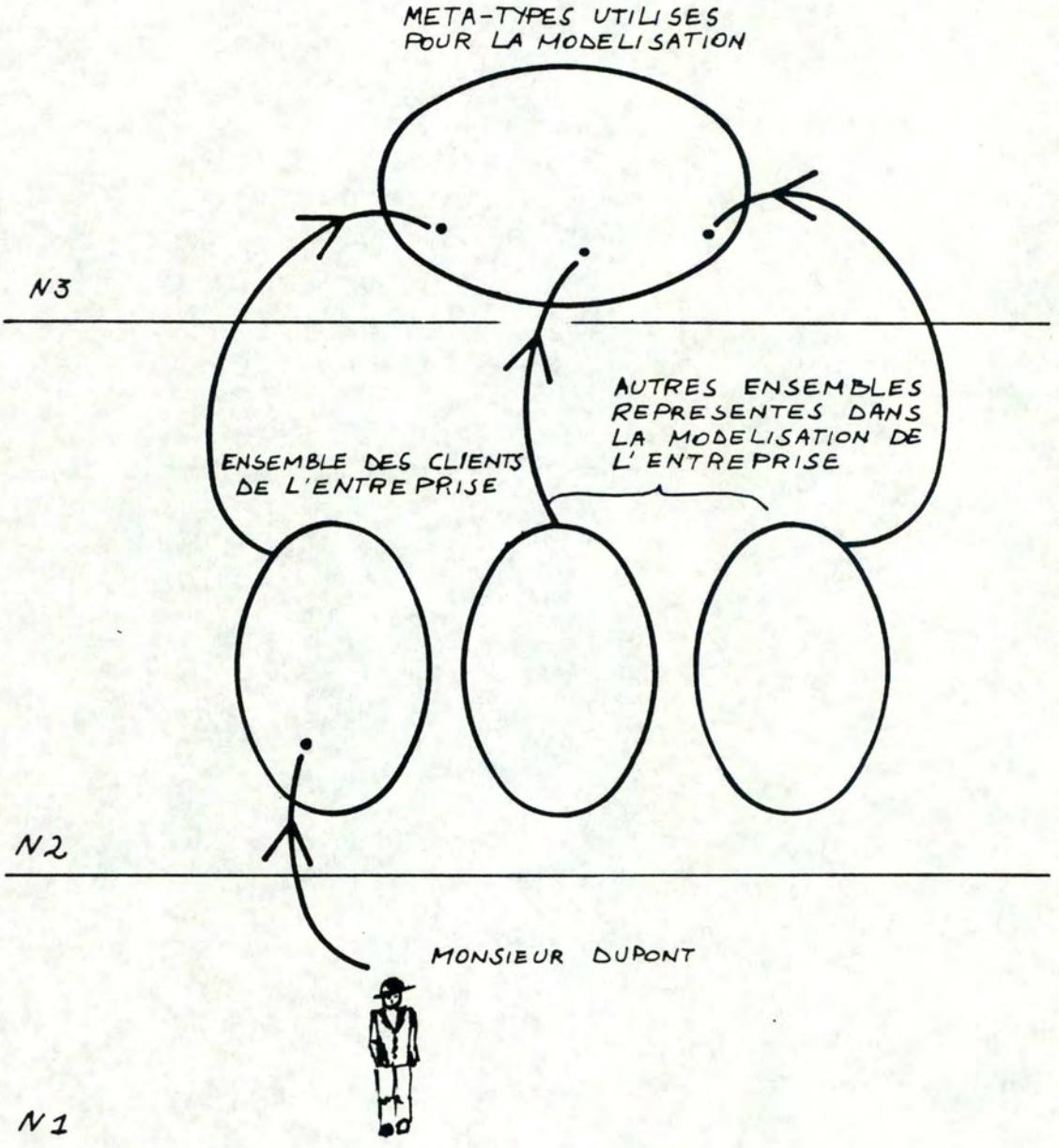
D'autre part, il contient des notions suffisamment générales pour etre utilisé comme modèle de description autonome, moins restrictif que le modèle E.A. (cette dernière idée devra etre nuancée).

Pour préciser ce second point, c'est-à-dire la manière dont le schéma est utilisé pour décrire une organisation ou un systéme, distinguons les trois niveaux d'abstraction en présence sur un exemple.

- M. DUPONT, client dans l'entreprise représente un élément du réel perçu.

- par abstraction, on considère l'ensemble des clients de l'entreprise comme le TYPE CLIENT et M. DUPONT en est une occurrence.

- par un mécanisme similaire, l'ensemble des types découverts représente un méta-type que l'on appellera type d'entité, d'association, d'article, de chemin suivant le modèle utilisé et le role qu'il joue par rapport à d'autres.


META-TYPES UTILISES
POUR LA MODELISATION
N3
AUTRES ENSEMBLES
REPRESENTES DANS
LA MODELISATION DE
L'ENTREPRISE
ENSEMBLE DES CLIENTS
DE L'ENTREPRISE
N2
MONSIEUR DUPONT
N1

Les composants de notre schéma représentent simultanément le niveau 2 et le niveau 3 :

niveau 3
---

Les concepts du schéma sont une généralisation de chaque type défini dans un schéma quelconque. Chacun de ceux-ci peut être un objet, une relation, une propriété et un élément.

niveau 2
---

Chaque occurrence dans une description est également une occurrence dans le schéma général.

Cette double appartenance d'une occurrence à plusieurs types dont certains sont une généralisation de certains autres traduit simplement la notion de surclasse.

Nous pouvons donc relier chaque composant d'un modèle quelconque par un lien "is a " vers un composant du schéma général. Il reste à préciser comment ces derniers se partagent les composants et comment ils sont eux-même organisés.

Nous allons les passer en revue pour trois modèles:

- le modèle E.A.
- le modèle d'accès généralisé (MAG)
- le modèle relationnel généralisé (MRG)

### 5.1.4.2. L'objet

La notion d'objet correspond exactement à un élément dans chacun des trois modèles. Leur définition précise très clairement cette notion.

Modéle E.A.
---

OBJET ≡ ENTITE

"...Une entité est une chose concréte ou abstraite appartenant au réel perçu à propos de laquelle on veut enregistrer des informations. Une entité n'existe en tant que telle que par rapport à un individu ou un groupe qui la considére comme un tout, lui confère une existance autonome et la distingue d'autres entités et de son environnement..." (Bodart)

Modéle MAG
---

OBJET ≡ ARTICLE

"...Chaque article est une unité d'information qui peut etre créée, supprimée et à laquelle on peut accéder. Chaque article est d'un type déterminé qui en définit les propriétés générales (items associés, identifiants, chemins d'accès, clés d'accès,..." (Hainaut)

Modèle MRG
---

OBJET ≡ DOMAINE ENTITE

"...Il apparait très clairement que parmi les domaines, certains désignent sans ambiguité des objets autonomes du monde réel (un contrat, un employé, un sinistre,...), (...).

Nous (les) appellerons (...) domaines-entités..." (Hainaut)

5.1.4.3. La propriété
---

Comme pour l'objet, la propriété correspond à un élément de chaque modéle. Elle est définie de la même façon.

Modéle E.A.
---

PROPRIETE ≡ ATTRIBUT

"...Un attribut peut etre considéré comme une relation qui fait correspondre à un élément d'un ensemble d'entité Ei ou d'un ensemble d'associations Ri:

- soit un élément d'un ensemble de valeurs (...)
- soit plusieurs éléments d'un même ensemble de valeurs (...)
- soit un ou plusieurs groupes d'éléments appartenant à plusieurs ensembles de valeurs..." (Bodart)

Modéle MAG
---

PROPRIETE ≡ VALEUR D'ITEM

"....Aux articles d'un type peuvent être associées des valeurs d'items. Une valeur d'item (l'item est défini comme un type de valeur) est une information se présentant comme une suite de symboles manipulables dans un programme. Il est entendu que l'on peut

obtenir la valeur d'item d'un article auquel on a accédé..." (Hainaut).

Modèle MRG
---

PROPRIETE ≡ DOMAINE-PROPRIETE

"...Il apparait clairement que parmi les domaines, certains désignent (...), tandis que d'autres (...) désignent des propriétés (des objets autonomes du monde réel). Nous appellerons (...) les seconds domaines-propriétés..." (Hainaut).

5.1.4.4. La relation
---

Pour la relation, les éléments des deux premiers modèles correspondent, mais nous devons etre plus précis pour le modèle MRG.

Modèle E.A.
---

RELATION ≡ ASSOCIATION

"...Une association est définie par une correspondance entre deux ou plusieurs entités (non nécessairement distinctes) où chacune assume un rôle donné. On veut enregistrer de l'information relative à cette correspondance. Une association peut posséder un ou plusieurs attributs..." (Bodart).

Modèle MAG
---

RELATION ≡ CHEMIN D'ACCES

"...Les articles peuvent être associés par des chemins d'accès inter-articles. Il s'agit d'un mécanisme qui associe à un article dit origine du chemin, des articles dits cibles de manière telle qu'il soit possible, à partir de l'origine, d'accéder dans un ordre déterminé aux cibles successives du chemin. Un chemin est unidirectionnel. Il appartient à un type qui en définit les propriétés générales (types origines et cibles, connectivité, contraintes d'intégrité, ..." (Hainaut).

Modèle MRG
---

RELATION ⊂ RELATION

"...Le monde réel n'est pas constitué d'ensembles isolés les uns des autres, mais d'éléments et propriétés en association mutuelle le dotant d'une structure généralement très riche (...). Nous

appellerons relation l'ensemble de toutes les associations de même nature..." (Hainaut).

Dans un schéma relationnel normalisé, on peut distinguer :

- les relations contenant au maximum un domaine-entité
- les relations contenant plusieurs domaines-entités.

Ces dernières traduisent notamment la correspondance entre ces domaines-entités. Cette seule partie de la sémantique d'une relation coïncide avec la relation du schéma général. De cette façon, cette dernière est incluse dans la première.

#### 5.1.4.5. L'élément et le texte associé

Pour donner une structure très riche au schéma, on introduit une surclasse par rapport aux trois concepts définis : l'élément. Tout objet, toute relation, toute propriété est aussi un élément, ce qui introduit un dernier niveau d'abstraction supérieur aux trois précédemment introduits.

L'introduction de l'élément va nous permettre d'économiser un grand nombre d'associations qui auraient été nécessaires entre :

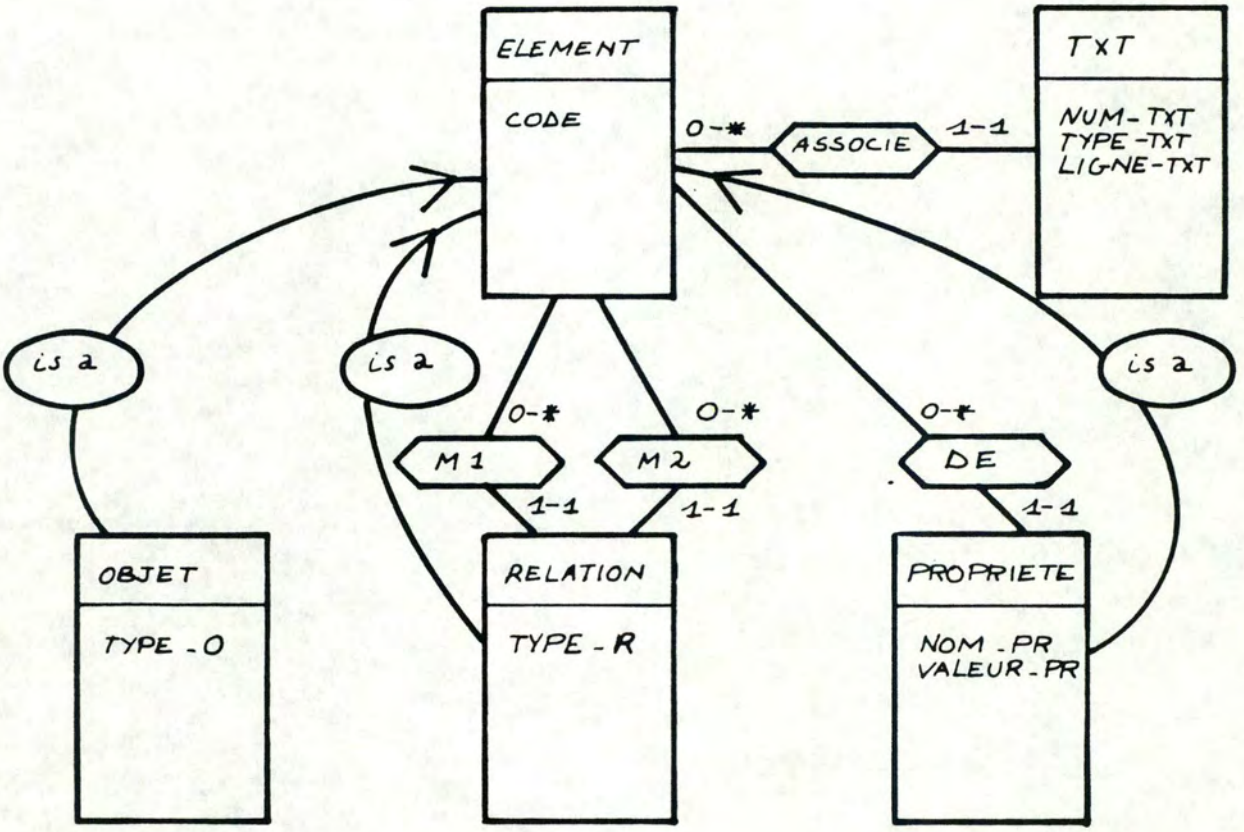
- objets et relations
- objets et attributs
- relations et attributs.

Celle-ci sont remplacées par des associations vers l'élément comme nous allons le voir.

Ceci nous permet également d'introduire un nouvel ensemble, le texte, dont chaque occurrence est une ligne de texte, dans laquelle on peut mettre ce que l'on veut et qui est associée à un objet, une relation ou une propriété de type quelconque, c'est-à-dire qu'elle est associée à l'élément.

#### 5.1.4.6. Organisation du schéma

Considérons plus particulièrement les notions et la terminologie du modèle E.A., le modèle le plus répandu.




L'association DE
---

permet de relier un attribut :

- à son T.E.
- à son T.A.
- à son attribut si celui-ci est décomposable

Les associations M1 et M2
---

permettent de relier une association aux deux T.E. sur lesquels elle porte, ceux-ci n'étant pas limités à cette seule association. Par contre, seules les associations binaires peuvent être représentées ici.

Ceci ajoute une contrainte par rapport au modèle E.A.

Le schéma permet également d'associer deux T.A. ou un T.E. et un T.A. Ceci lève une contrainte du modèle E.A. et peut etre utile. Il permet en outre les associations entre un élément quelconque et un attribut, ce qui, par contre, n'a pas de sens.

Le type de l'objet et de la relation

est leur seul attribut propre; il définit et identifie la sous-classe particulière dans la classe objet ou relation c'est-à-dire que dans un modèle E.A., chaque T.E. et chaque T.A. auront un nom distinct.

Le nom et la valeur de la propriété

permet de représenter tous les autres attributs associés à chaque objet et relation. Parmi les objets d'un type donné, le nom de la propriété identifie la classe de valeurs de cette propriété, c'est-à-dire que dans le modèle E.A., chaque attribut d'un T.E. ou d'un T.A. possédera un nom distinct des autres attributs de ce meme T.E. ou T.A.

L'identifiant de l'élément

est le code de l'élément quel qu'il soit, la structure arborescente permet au code d'identifier également l'objet, la relation et la propriété.

L'identifiant de l'objet

outre le code de l'élément, l'objet peut éventuellement posséder un identifiant propre à un type particulier par exemple le nom de l'objet; dans ce cas, un second identifiant sera le nom et le type.

L'identifiant de la relation

peut etre triple car la remarque faite pour l'objet s'applique ici avec , en outre, un troisième identifiant : le code de l'élément via M1, le code de l'élément via M2 et le type de la relation.

L'identifiant de la propriété

peut etre triple également, le troisième identifiant est constitué du code de l'élément via DE, le nom et la valeur de la propriété, solution valable seulement si la propriété est non répétitive.

L'identifiant du texte
---

est le numéro de la ligne et le code de l'élément via ASSOCIE

## 5.1.5. EXEMPLE D'UTILISATION

### 5.1.5.1. Schema classique

En conservant le formalisme E.A., considérons l'exemple classique de la passation d'une commande par des clients.

Schéma conceptuel

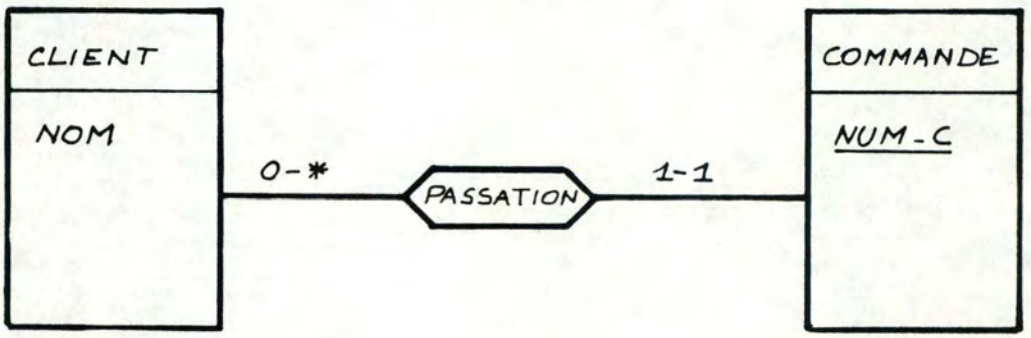
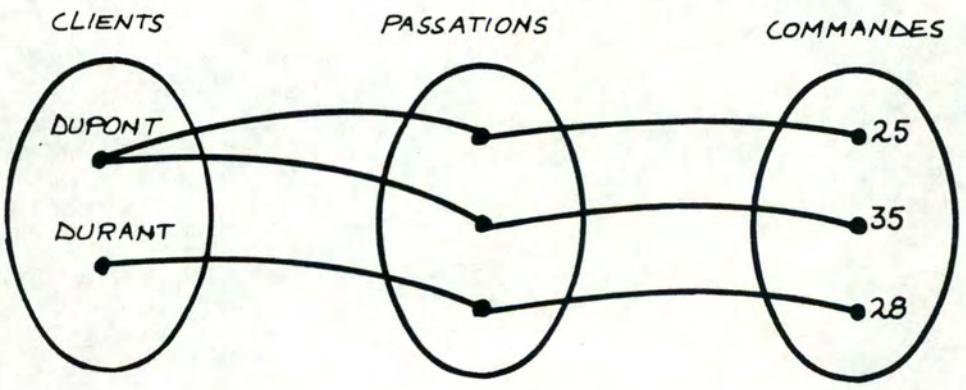



Schéma d'occurrences

supposons que :

- M. DUPONT a passé les commandes n° : 25 et 35
- M. DURANT a passé la commande n° : 28



### 5.1.5.2. Schéma général

La traduction du même exemple dans le schéma général nous oblige à ajouter un code identifiant pour chaque élément :

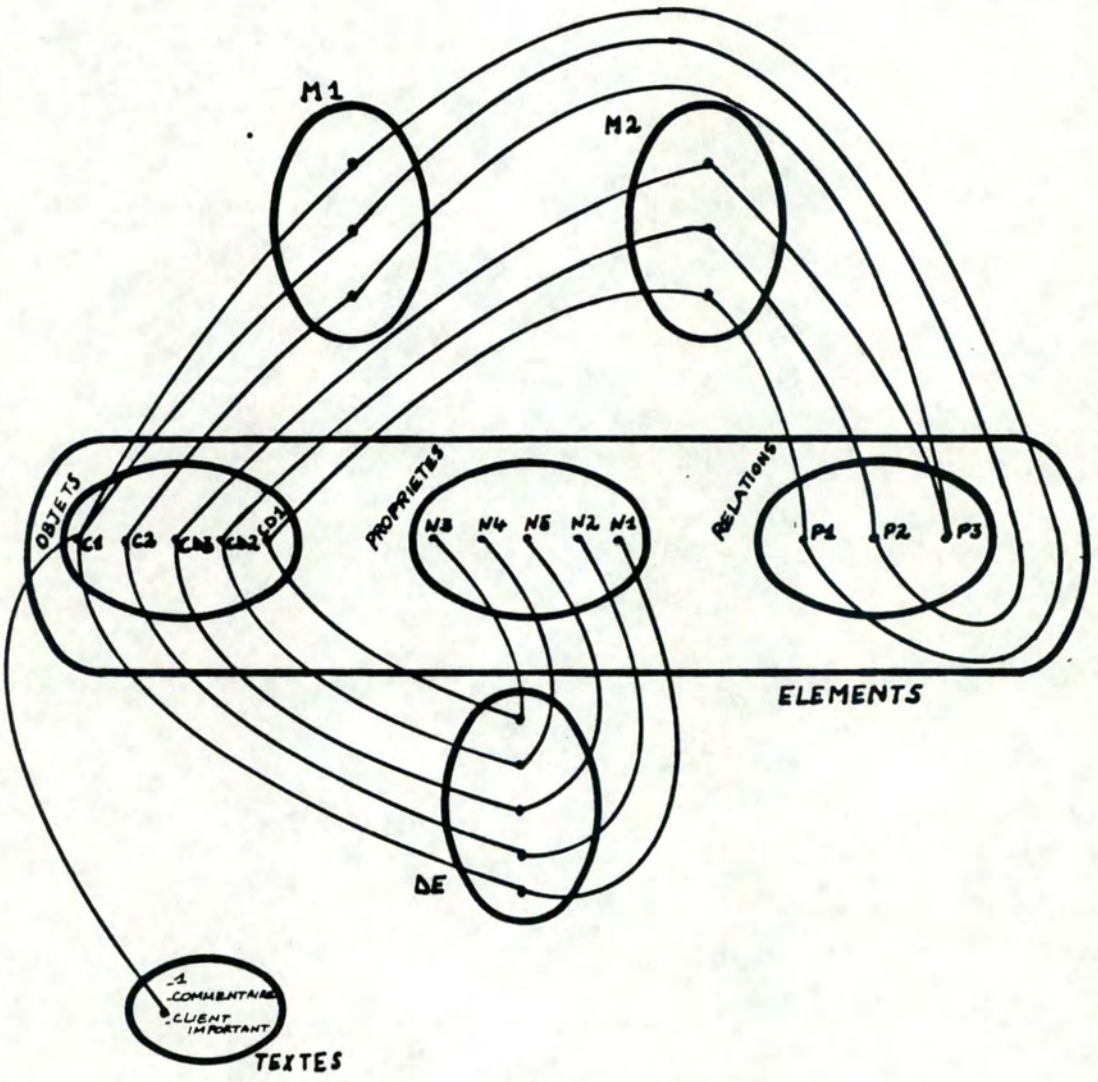
clients
- M.Dupont : C1
- M.Durant : C2

commandes
- commande n° 25 : CD1
- commande n° 35 : CD2
- commande n° 28 : CD3

passations
- de la commande n° 25 par M.Dupont : P1
- de la commande n° 35 par M.Dupont : P2
- de la commande n° 28 par M.Durand : P3

attributs
- nom de M.Dupont : N1
- nom de M.Durand : N2
- num-c de la commande n° 25 : N3
- num-c de la commande n° 35 : N4
- num-c de la commande n° 28 : N5

Schéma d'occurrences



### 5.1.6. CONCLUSION

Outre l'intéret de son étude théorique, le schéma général présente l'avantage d'une très grande souplesse et d'une indépendance vis-à-vis d'un modèle.

Par contre, il necessite l'introduction d'un code identifiant généralement redondant pour chaque élément, il ne peut traduire toutes les notions de certains modèles, et le nombre d'occurrences croit très rapidement comme l'indique le dernier schéma.

# ANNEXE 5.2.

# TRANSFORMATION : MODELE E.A. EN MODELE MAG

## 5.2.1. INTRODUCTION

Nous ne développerons pas ici la théorie complète de transformation du modèle E.A. en modéle MAG. (Le lecteur intéressé trouvera cette description complète dans 13). Nous nous limiterons à l'exposé des grands points qui nous sont nécessaires pour la transformation du schéma conceptuel du dictionnaire de données (que nous appellerons le noyau) en schéma MAG des accès possibles c'est-à-dire sans se préoccuper des accès nécessaires à chaque requete.

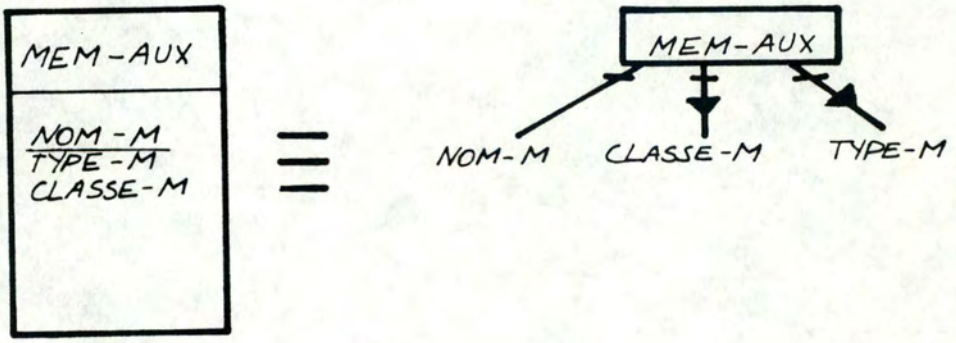
Ce dernier schéma est utilisé pour l'implémentation du noyau (chapitre 5.5.5) qui impose des restrictions supplémentaires au schéma MAG obtenu; nous les considérons de meme que leurs conséquences sur le schéma conceptuel de départ. (Ce raisonnement peu conventionnel est du à un choix d'implémentation (chapitre 5.5.5.1)).

Nous citerons pour terminer les restrictions à appliquer sur ce schéma MAG pour le rendre compatible avec les SGBD relationnels.

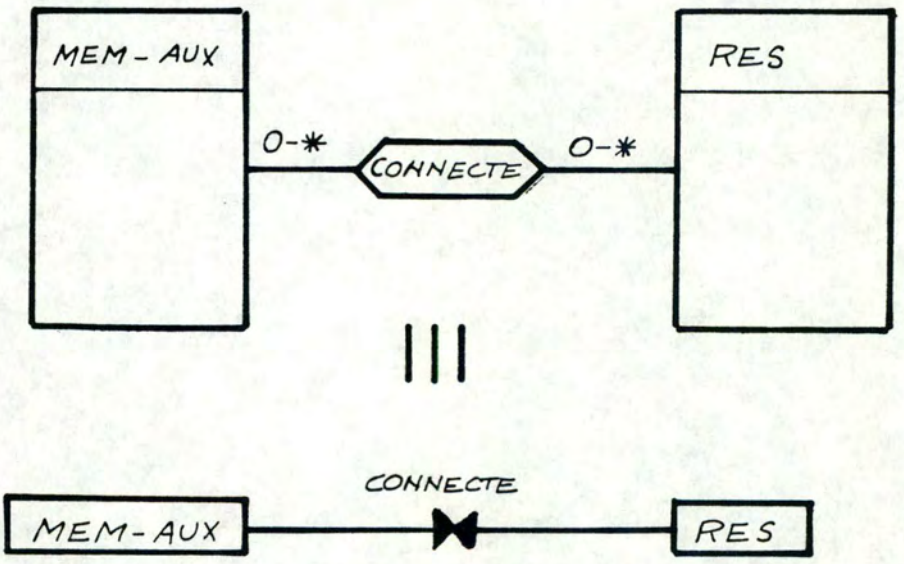
## 5.2.2. REGLES DE TRANSFORMATION

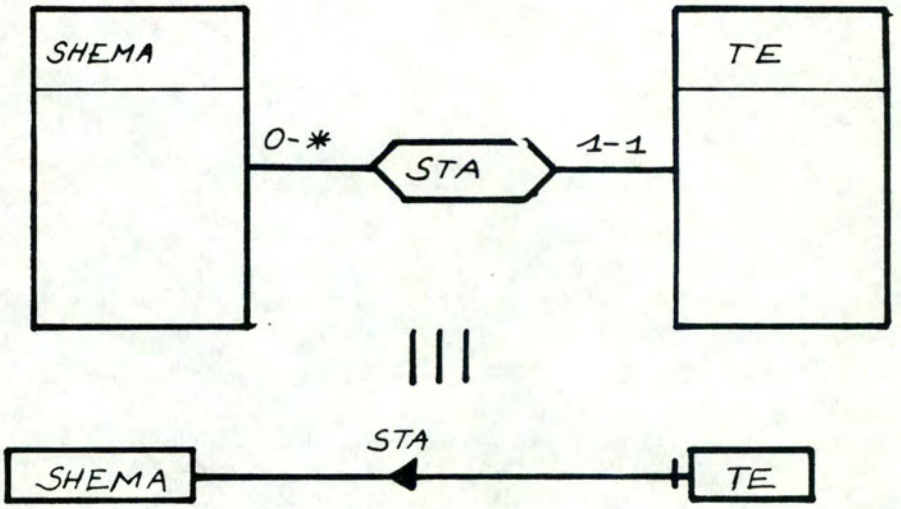
Un type d'entité et ses attributs devient un type d'article et ses items .

Exemple :

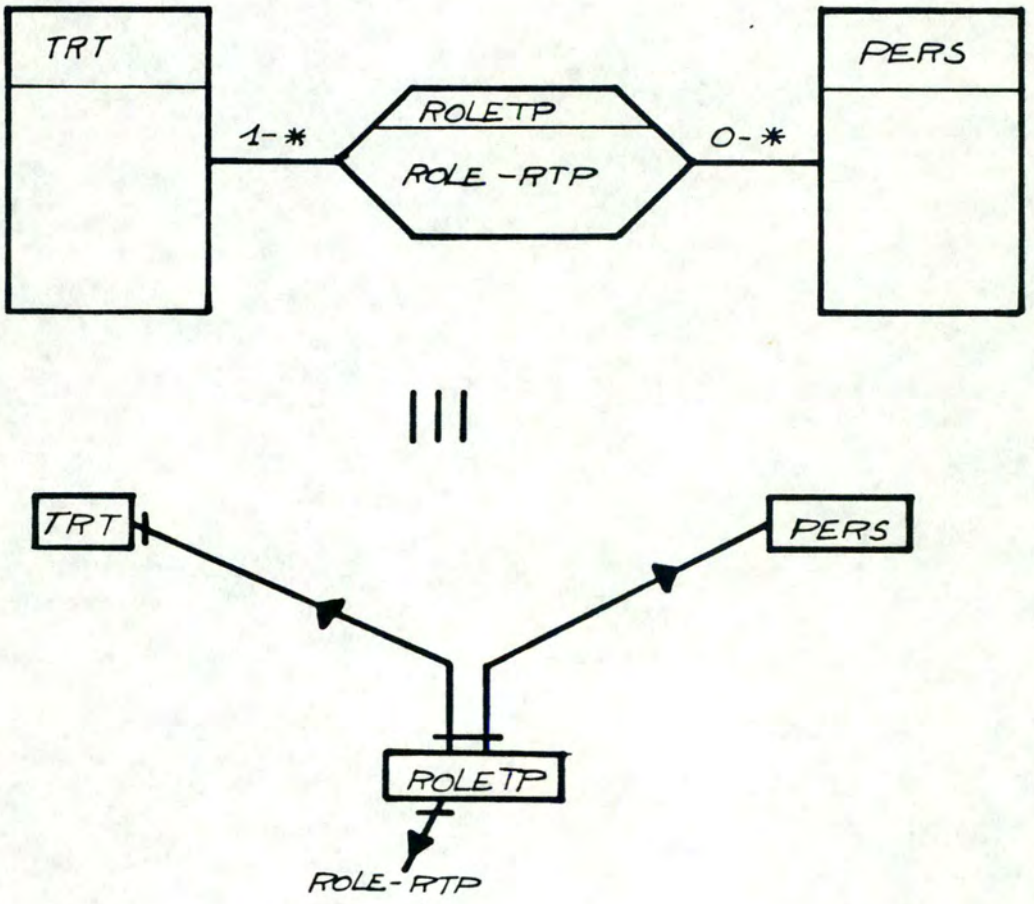



Un type d'association binaire sans attributs devient un type de chemin.






Un type d'association binaire avec attributs se transforme en :
- un type d'article
- deux types de chemin 1-N
- deux contraintes d'existence sur les types de chemins
- le type d'article est identifié par les deux types de chemins.



### 5.2.3. RESTRICTIONS SUPPLEMENTAIRES SUR LE SCHEMA MAG

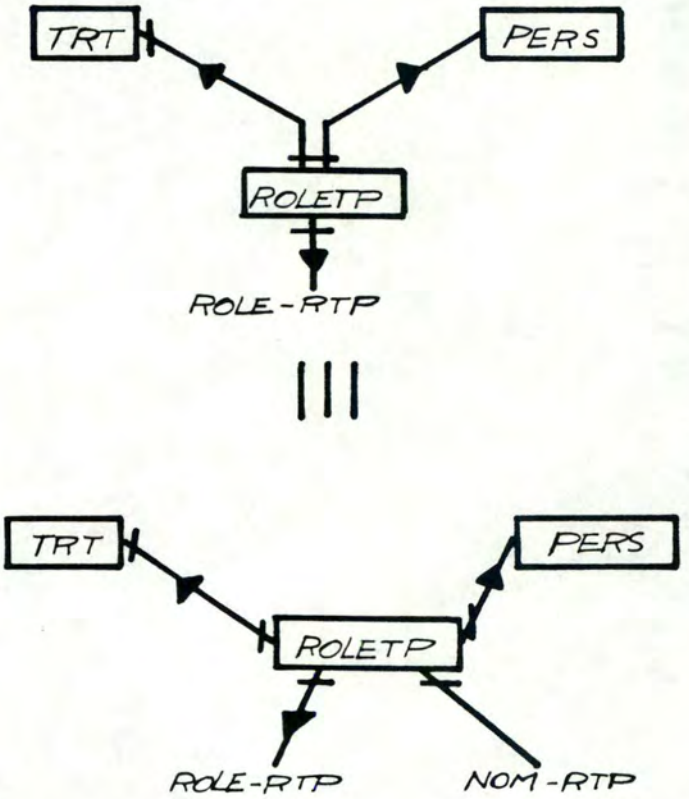
L'implémentation du DD par une structure intermédiaire nous impose une restriction supplémentaire. Tout type d'article est identifié par un seul item qui lui est propre.

Conséquences :

Tout type d'article issu d'un T.E. du schéma conceptuel du noyau sera identifié par son nom. Ceci modifie les spécifications des types d'entité : T.E., TAS, IKO, ATTR.

A tout type d'article issu d'un T.A. avec attributs, on ajoute un item NOM (sauf si le T.A. possédait déjà cet attribut) qui devient l'identifiant.

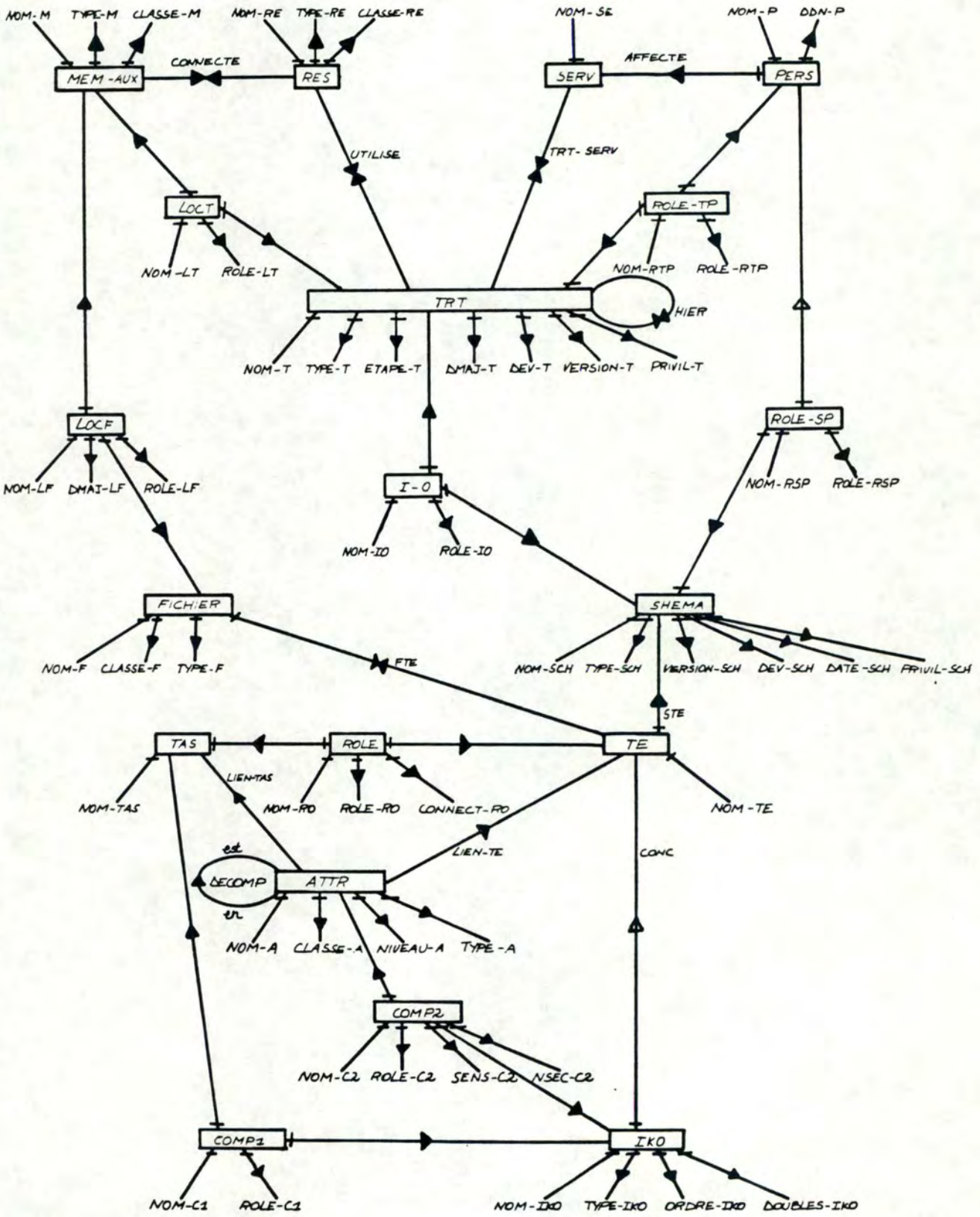
En reprenant l'exemple précédent, nous obtenons la transformation suivante :




Contrainte:

NOM-RTP traduit la contrainte définie par les deux types de chemins. Une convention pourrait être que NOM-RTP est la concaténation des noms du traitement et de la personne.

## 5.2.4. Schéma MAG. transformé.



### 5.2.5. SCHEMAS MAG COMPATIBLE SQL.

Nous prenons la définition dans(16):
"...Par rapport aux schémas d'accès généraux, les schémas relationnels se caractérisent par un ensemble de restrictions dont on rapelle les principales :

1) pas de types de chemins,
2) pas d'items décomposables,
3) pas d'items répétitifs,
4) pas d'items facultatifs (sauf possibilité de NULL VALUE en SQL),
5) au moins un item par type d'articles,
6) au moins un identifiant par type d'articles..."

(Hainaut)

# ANNEXE 5.3

## LISTE DES TABLES ET VUES CREES

```
1. create table OBJET
        (TYPE-O          varchar (12)    not null,
         NOM-O           varchar (12)    not null,
         SSTYPE-O        varchar (12),
         SSCLASSE-O      varchar (12),
         DATE-O          integer,
         C6-O            varchar (12),
         C7-O            smallint,
         C8-O            varchar (2))
                in STUDENT

2. create table RELATION
        (TYPE1-R         varchar (12)    not null,
         NOM1-R          varchar (12)    not null,
         TYPE2-R         varchar (12)    not null,
         NOM2-R          varchar (12)    not null,)
                in STUDENT

3. create table TEXTE
        (TYPE-OAS        varchar (12)    not null,
         NOM-OAS         varchar (12)    not null,
         NUML-TXT        integer         not null,
         TYPE-TXT        varchar (12),
         LIGNE-TXT       varchar (80))
                in STUDENT

4. create table PROPRIETE
        (TYPE-DEPR       varchar (12)    not null,
         NOM-DEPR        varchar (12)    not null,
         NOM-PR          varchar (12)    not null,
         VALEUR-PR       varchar (12))
                in STUDENT

5. create unique index
        IDX-OBJET on OBJET          (TYPE-O     asc,
                                     NOM-O      asc)

6. create unique index
        IDX-RELATION on RELATION (TYPE1-R    asc,
                                  TYPE2-R    asc,
                                  NOM1-R     asc,
                                  NOM2-R     asc)
```

```
7. create unique index
        IDX-TEXTE on TEXTE          (TYPE-OAS   asc,
                                     NOM-OAS    asc,
                                     NUML-OAS   asc)


8. create unique index
        IDX-PROPRIETE on PROPRIETE (TYPE-DEPR    asc,
                                    NOM-DEPR     asc,
                                    NOM-PR       asc,
                                    VALEUR-PR    asc)

9. create view PERS
            (NOM-P, DDN-P, NOM-SEP) as
           select
              NOM-O, DATE-O, NOM2-R
           from
              OBJET, RELATION
           where
              TYPE-O = TYPE1-R    and
              TYPE1-R = 'PERS'    and
              NOM-O = NOM1-R      and
              TYPE2-R = 'SERV'

10. create view SERV
              (NOM-SE) as
            select
              NOM-O
            from
              OBJET
            where
              TYPE-O = 'SERV'

11. create views TRT
              (NOM-T, TYPE-T, ETAPE-T, DEV-T, DMAJ-T,
               VERS-T, PRIVIL-T) as
            select
               NOM-O, SSTYPE-O, SSCLASSE-O, C6-O, C7-O,
               C8-O
            from
               OBJET
            where
               TYPE-O = 'TRT'
```

```
12. create view SCHEMA
      (NOM-SCH, TYPE-SCH, VERS-SCH, DEV-SCH,
       DATE-SCH, PRIVIL-SCH) as
    select
       NOM-O, SSTYPE-O, C7-O, C6-O, DATE-O,
       C8-O
    from
       OBJET
    where
       TYPE-O = 'SCHEMA'

13. create view TE
      (NOM-TE, NOM-SCHTE) as
    select
       NOM-O, NOM1-R
    from
       OBJET, RELATION
    where
       TYPE-O = 'TE'       and
       TYPE2-R = TYPE-O    and
       NOM-O = NOM2-R      and
       TYPE1-R = 'SCHEMA'

14. create view ATTR
      (NOM-A, CLASSE-A, NIVEAU-A, TYPE-A,
       NOM-LIEN, TYPE-LIEN)  as
    select
       NOM-O, SSCLASSE-O, C7-O, SSTYPE-O,
       NOM2-R, TYPE2-R
    from
       OBJET, RELATION
    where
       TYPE-O = 'ATTR'          and
       TYPE1-R = TYPE-O         and
       TYPE2-R in ('TE','TAS') and
       NOM-O = NOM1-R.

15. create view IKO
      (NOM-IKO, TYPE-IKO, ORDRE-IKO, DOUBLES-IKO
       NOM-TEIKO)  as
    select
       NOM-O, SSTYPE-O, C6-O, C8-O, NOM2-R
    from
       OBJET, RELATION
    where
       TYPE-O = 'IKO'           and
       TYPE1-R = TYPE-O         and
       TYPE2-R = 'TE'           and
       NOM-O = NOM1-R
```

```
16. create view TAS
      (NOM-TAS)  as
    select
       NOM-O
    from
       OBJET
    where
       TYPE-O = 'TAS'

17. create view FICHIER
      (NOM-F, CLASSE-F, TYPE-F)  as
    select
       NOM-O, SSCLASSE-O, SSTYPE-O
    from
       OBJET
    where
       TYPE-O = 'FICHIER'

18. create view MEM-AUX
      (NOM-M, TYPE-M, CLASSE-M)  as
    select
       NOM-O, SSTYPE-O, SSCLASSE-O
    from
       OBJET
    where
       TYPE-O = 'MEM-AUX'

19. create view RES
      (NOM-RE, TYPE-RE, CLASSE-RE)  as
    select
       NOM-O, SSTYPE-O, SSCLASSE-O
    from
       OBJET
    where
       TYPE-O = 'RES'

20. create view LOCT
      (NOM-LT, ROLE-LT, NOM-MLT, NOM-TLT)  as
    select
       NOM-O, C6-O, R1.NOM2-R, R2.NOM2-R
    from
       OBJET, RELATION R1, RELATION R2
    where
       TYPE-O = 'LOCT'          and
       R1.TYPE1-R = TYPE-O      and
       R2.TYPE1-R = TYPE-O      and
       NOM-O = R1.NOM1-R        and
       NOM-O = R2.NOM1-R        and
       R1.TYPE2-R = 'SCHEMA'    and
       R2.TYPE2-R = 'TRT'
```

```
21. create view LOCF
      (NOM-LF, DMAJ-LF, ROLE-LF, NOM-MLF,
       NOM-FLF)  as
    select
       NOM-O, DATE-O, C6-O, R1.NOM2-R, R2.NOM1-R
    from
       OBJET, RELATION R1, RELATION R2
    where
       TYPE-O = 'LOCF'          and
       R1.TYPE1-R = NOM-O       and
       R2.TYPE2-R = TYPE-O      and
       NOM-O = R1.NOM1-R        and
       NOM-O = R2.NOM2-R        and
       R1.TYPE2-R = 'MEM-AUX'   and
       R2.TYPE1-R = 'FICHIER'

22. create view ROLETP
      (NOM-RTP,ROLE-RTP, NOM-TRTP, NOM-PRTP)  as
    select
       NOM-O, C6-O, R1.NOM2-R, R2.NOM1-R
    from
       OBJET, RELATION R1, RELATION R2
    where
       TYPE-O = 'ROLETP'        and
       R1.TYPE1-R = TYPE-O      and
       R2.TYPE2-R = TYPE-O      and
       NOM-O = R2.NOM2-R        and
       R1.NOM1-R = NOM-O        and
       R1.TYPE2-R = 'TRT'       and
       R2.TYPE1-R = 'PERS'

23. create view ROLESP
      (NOM-RSP, ROLE-RSP, NOM-SCHRSP, NOM-PRSP)
       as
    select
       NOM-O, C6-O, R1.NOM2-R, R2.NOM1-R
    from
       OBJET, RELATION R1, RELATION R2
    where
       TYPE-O = 'ROLESP'        and
       R1.TYPE1-R = TYPE-O      and
       R2.TYPE2-R = TYPE-O      and
       NOM-O = R1.NOM1-R        and
       NOM-O = R2.NOM2-R        and
       R1.TYPE2-R = 'SCHEMA'    and
       R2.TYPE1-R = 'PERS'
```

```
24. create view I-O
      (NOM-IO, ROLE-IO, NOM-TIO, NOM-SCHIO)  as
    select
       NOM-O, C6-O, R1.NOM2-R, R2.NOM2-R
    from
       OBJET, RELATION R1, RELATION R2
    where
       TYPE-O = 'I-O'              and
       R1.TYPE1-R = TYPE-O         and
       R2.TYPE1-R = TYPE-O         and
       NOM-O = R1.NOM1-R           and
       NOM-O = R2.NOM1-R           and
       R1.TYPE2-R = 'TRT'          and
       R2.TYPE2-R = 'SCHEMA'

25. create view ROLE
      (NOM-RO, ROLE-RO, CONNECT-RO,NOM-TASRO,
       NOM-TERO)  as
    select
       NOM-O, C6-O, C8-O, R1.NOM2-R, R2.NOM2-R
    from
       OBJET, RELATION R1, RELATION R2
    where
       TYPE-O = 'ROLE'             and
       R1.TYPE1-R = TYPE-O         and
       R2.TYPE1-R = TYPE-O         and
       NOM-O = R1.NOM1-R           and
       NOM-O = R2.NOM1-R           and
       R1.TYPE2-R = 'TAS'          and
       R2.TYPE2-R = 'TE'

26. create view COMP1
      (NOM-C1, ROLE-C1, NOM-IKOC1, NOM-TASC1)
       as
    select
       NOM-O, C6-O, R1.NOM2-R, R2.NOM2-R
    from
       OBJET, RELATION R1, RELATION R2
    where
       TYPE-O = 'COMP1'            and
       R1.TYPE1-R = TYPE-O         and
       R2.TYPE1-R = TYPE-O         and
       NOM-O = R1.NOM1-R           and
       NOM-O = R2.NOM1-R           and
       R1.TYPE2-R = 'IKO'          and
       R2.TYPE2-R = 'TAS'
```

```
27. create viev COMP2
      (NOM-C2, ROLE-C2, SENS-C2, NSEQ-C2,
       NOM-IKOC2, NOM-AC2)  as
    select
       NOM-O, C6-O, C8-O, C7-O, R1.NOM2-R,
       R2.NOM1-R
    from
       OBJET, RELATION R1, RELATION R2
    where
       TYPE-O = 'COMP2'          and
       R1.TYPE1-R = TYPE-O       and
       R2.TYPE2-R = TYPE-O       and
       NOM-O = R1.NOM1-R         and
       NOM-O = R2.NOM2-R         and
       R1.TYPE2-R = 'IKO'        and
       R2.TYPE1-R = 'ATTR'

28. create view CONNECTE
      (NOM-MC, NOM-REC)  as
    select
       NOM1-R, NOM2-R
    from
       RELATION
    where
       TYPE1-R = 'MEM-AUX'       and
       TYPE2-R = 'RES'

29. create view UTILISE
      (NOM-REU, NOM-TU)  as
    select
       NOM1-R, NOM2-R
    from
       RELATION
    where
       TYPE1-R = 'RES'           and
       TYPE2-R = 'TRT'

30. create view TRT-SERV
      (NOM-TT, NOM-SET)  as
    select
       NOM2-R, NOM1-R
    from
       RELATION
    where
       TYPE2-R = 'TRT'           and
       TYPE1-R = 'SEV'
```

```
31. create view FTE
              (NOM-FF, NOM-TEF)  as
            select
               NOM1-R, NOM2-R
            from
               RELATION
            where
               TYPE1-R = 'FICHIER'      and
               TYPE2-R = 'TE'

32. create view DECOMP
              (NOM-APD, NOM-AFD)  as
            select
               NOM2-R, NOM1-R
            from
               RELATION
            where
               TYPE1-R = 'ATTR'         and
               TYPE2-R = 'ATTR'

33. create view HIER
              (NOM-TPH, NOM-TFH)  as
            select
               NOM2-R, NOM1-R
            from
               RELATION
            where
               TYPE1-R = 'TRT'          and
               TYPE2-R = 'TRT'
```

# ANNEXE 7.1.

## LES TABLES: OBJET ET RELATION

| TYPE_O | NOM_O | SSTYPE_O | SSCLASSE_O | DATE_O | C6_O | C7_O | C8_O |
|---|---|---|---|---|---|---|---|
| ATTR | ABF | CHAR | FES | ? | ? | ? | ? |
| | ABL | CHAR | FES | ? | ? | ? | ? |
| | ACR | CHAR | FES | ? | ? | ? | ? |
| | ADR | CHAR | FES | ? | ? | ? | ? |
| | AFF | CHAR | FES | ? | ? | ? | ? |
| | ANR | CHAR | FES | ? | ? | ? | ? |
| | ARC | CHAR | FES | ? | ? | ? | ? |
| | BTE | CHAR | FES | ? | ? | ? | ? |
| | BUR | CHAR | FES | ? | ? | ? | ? |
| | CAD | CHAR | FES | ? | ? | ? | ? |
| | CAO | CHAR | FES | ? | ? | ? | ? |
| | CAT | CHAR | FES | ? | ? | ? | ? |
| | CEA | CHAR | FES | ? | ? | ? | ? |
| | CFA | CHAR | FES | ? | ? | ? | ? |
| | CFC | CHAR | FES | ? | ? | ? | ? |
| | CIO | CHAR | FES | ? | ? | ? | ? |
| | CJT | CHAR | FES | ? | ? | ? | ? |
| | CLASSE_A | VCHAR12 | FES | ? | ? | 1 | ? |
| | CLASSE_F | VCHAR12 | FES | ? | ? | 1 | ? |
| | CLASSE_M | VCHAR12 | FES | ? | ? | 1 | ? |
| | CLASSE_RE | VCHAR12 | FES | ? | ? | 1 | ? |
| | CMP | CHAR | FES | ? | ? | ? | ? |
| | CON | CHAR | FES | ? | ? | ? | ? |
| | CONNECT_RO | VARCHAR2 | FES | ? | ? | 1 | ? |
| | CRE | CHAR | FES | ? | ? | ? | ? |
| | CSA | CHAR | FES | ? | ? | ? | ? |
| | CST | CHAR | FES | ? | ? | ? | ? |
| | CZR | CHAR | FES | ? | ? | ? | ? |
| | DATE_SCH | INTEGER | FES | ? | ? | 1 | ? |
| | DCR | CHAR | FES | ? | ? | ? | ? |
| | DON_P | INTEGER | FES | ? | ? | 1 | ? |
| | DEA | CHAR | FES | ? | ? | ? | ? |
| | DEV_SCH | VCHAR12 | FES | ? | ? | 1 | ? |
| | DEV_T | VCHAR12 | FES | ? | ? | 1 | ? |
| | DIG | CHAR | FES | ? | ? | ? | ? |
| | DIP | CHAR | FES | ? | ? | ? | ? |
| | DMAJ_LF | INTEGER | FES | ? | ? | 1 | ? |
| | DMAJ_T | INTEGER | FES | ? | ? | 1 | ? |
| | DMJ | CHAR | FES | ? | ? | ? | ? |
| | DOUBLES_IKO | VARCHAR2 | FES | ? | ? | 1 | ? |
| | DRA | CHAR | FES | ? | ? | ? | ? |
| | DSA | CHAR | FES | ? | ? | ? | ? |
| | ECI | CHAR | FES | ? | ? | ? | ? |
| | ENR | CHAR | FES | ? | ? | ? | ? |
| | ETAPE_T | VCHAR12 | FES | ? | ? | 1 | ? |
| | GEN | CHAR | FES | ? | ? | ? | ? |
| | INA | CHAR | FES | ? | ? | ? | ? |
| | INS | CHAR | FES | ? | ? | ? | ? |
| | JNA | CHAR | FES | ? | ? | ? | ? |
| | LOC | CHAR | FES | ? | ? | ? | ? |
| | MUC | CHAR | FES | ? | ? | ? | ? |
| | NAC | CHAR | FES | ? | ? | ? | ? |
| | NAT | CHAR | FES | ? | ? | ? | ? |
| | NIVEAU_A | SMALLINT | FES | ? | ? | 1 | ? |
| | NNA | CHAR | FES | ? | ? | ? | ? |
| | NOF | CHAR | FES | ? | ? | ? | ? |
| | NOL | CHAR | FES | ? | ? | ? | ? |
| | NOM_A | VCHAR12 | OES | ? | ? | 1 | ? |

| TYPE_O | NOM_O | SSTYPE_O | SSCLASSE_O | DATE_O | C6_O | C7_O | C8_O |
|---|---|---|---|---|---|---|---|
| ATTR | NOM_C1 | VARCHAR12 | OES | ? | ? | 1 | ? |
| | NOM_C2 | VARCHAR12 | OES | ? | ? | 1 | ? |
| | NOM_F | VCHAR12 | OES | ? | ? | 1 | ? |
| | NOM_IKO | VARCHAR12 | OES | ? | ? | 1 | ? |
| | NOM_IO | VARCHAR12 | OES | ? | ? | 1 | ? |
| | NOM_LF | VARCHAR12 | OES | ? | ? | 1 | ? |
| | NOM_LT | VARCHAR12 | OES | ? | ? | 1 | ? |
| | NOM_M | VCHAR12 | OES | ? | ? | 1 | ? |
| | NOM_P | VCHAR12 | OES | ? | ? | 1 | ? |
| | NOM_RE | VCHAR12 | OES | ? | ? | 1 | ? |
| | NOM_RO | VARCHAR12 | OES | ? | ? | 1 | ? |
| | NOM_RSP | VARCHAR12 | OES | ? | ? | 1 | ? |
| | NOM_RTP | VARCHAR12 | OES | ? | ? | 1 | ? |
| | NOM_SCH | VCHAR12 | OES | ? | ? | 1 | ? |
| | NOM_SE | VCHAR12 | OES | ? | ? | 1 | ? |
| | NOM_T | VCHAR12 | OES | ? | ? | 1 | ? |
| | NOM_TAS | VCHAR12 | OES | ? | ? | 1 | ? |
| | NOM_TE | VCHAR12 | OES | ? | ? | 1 | ? |
| | NUM | CHAR | OES | ? | ? | ? | ? |
| | ORDRE_IKO | VARCHAR12 | FES | ? | ? | 1 | ? |
| | ORR | CHAR | FES | ? | ? | ? | ? |
| | PAN | CHAR | FES | ? | ? | ? | ? |
| | PHO | CHAR | FES | ? | ? | ? | ? |
| | PLC | CHAR | FES | ? | ? | ? | ? |
| | PPE | CHAR | FES | ? | ? | ? | ? |
| | PRF | CHAR | FES | ? | ? | ? | ? |
| | PRIV_SCH | VCHAR2 | FES | ? | ? | 1 | ? |
| | PRIV_T | VCHAR2 | FES | ? | ? | 1 | ? |
| | PRO | CHAR | FES | ? | ? | ? | ? |
| | QUA | CHAR | FES | ? | ? | ? | ? |
| | RAF | CHAR | FES | ? | ? | ? | ? |
| | REV | CHAR | FES | ? | ? | ? | ? |
| | RLI | CHAR | FES | ? | ? | ? | ? |
| | ROLE_C1 | VARCHAR12 | FES | ? | ? | 1 | ? |
| | ROLE_C2 | VARCHAR12 | FES | ? | ? | 1 | ? |
| | ROLE_IO | VARCHAR12 | FES | ? | ? | 1 | ? |
| | ROLE_LF | VARCHAR12 | FES | ? | ? | 1 | ? |
| | ROLE_LT | VARCHAR12 | FES | ? | ? | 1 | ? |
| | ROLE_RO | VARCHAR12 | FES | ? | ? | 1 | ? |
| | ROLE_RSP | VARCHAR12 | FES | ? | ? | 1 | ? |
| | ROLE_RTP | VARCHAR12 | FES | ? | ? | 1 | ? |
| | RRP | CHAR | FES | ? | ? | ? | ? |
| | SEA | CHAR | FES | ? | ? | ? | ? |
| | SENS_C1 | VARCHAR2 | FES | ? | ? | 1 | ? |
| | SENS_C2 | VARCHAR2 | FES | ? | ? | 1 | ? |
| | SEX | CHAR | FES | ? | ? | ? | ? |
| | SRP | CHAR | FES | ? | ? | ? | ? |
| | SSA | CHAR | FES | ? | ? | ? | ? |
| | TEA | CHAR | FES | ? | ? | ? | ? |
| | TSA | CHAR | FES | ? | ? | ? | ? |
| | TYPE_A | VARCHAR12 | FES | ? | ? | 1 | ? |
| | TYPE_F | VCHAR12 | FES | ? | ? | 1 | ? |
| | TYPE_IKO | VARCHAR12 | FES | ? | ? | 1 | ? |
| | TYPE_M | VCHAR12 | FES | ? | ? | 1 | ? |
| | TYPE_RE | VCHAR12 | FES | ? | ? | 1 | ? |
| | TYPE_SCH | VCHAR12 | FES | ? | ? | 1 | ? |
| | TYPE_T | VCHAR12 | FES | ? | ? | 1 | ? |
| | VERS_SCH | SMALLINT | FES | ? | ? | 1 | ? |

| TYPE_O | NOM_O | SSTYPE_O | SSCLASSE_O | DATE_O | Co_O | C7_O | C8_O |
|---|---|---|---|---|---|---|---|
| ATTR | VERS_T | SMALLINT | FES | ? | ? | 1 | ? |
| FICHIER | AFFILI | TABLESQL | PERM | ? | ? | ? | ? |
| I_O | INSATTR1DD | ? | ? | ? | INSERT | ? | ? |
| | INSATTR2DD | ? | ? | ? | INSERT | ? | ? |
| | INSTASDD | ? | ? | ? | INSERT | ? | ? |
| | INSTEDD | ? | ? | ? | INSERT | ? | ? |
| | INSTRT1DD | ? | ? | ? | INSERT | ? | ? |
| | INSTXTDD | ? | ? | ? | INSERT | ? | ? |
| IKO | IA | I | ? | ? | ? | ? | ? |
| | IF | I | ? | ? | ? | ? | ? |
| | IIKO | I | ? | ? | ? | ? | ? |
| | IM | I | ? | ? | ? | ? | ? |
| | IP | I | ? | ? | ? | ? | ? |
| | IRE | I | ? | ? | ? | ? | ? |
| | ISCH | I | ? | ? | ? | ? | ? |
| | ISE | I | ? | ? | ? | ? | ? |
| | IT | I | ? | ? | ? | ? | ? |
| | ITAS | I | ? | ? | ? | ? | ? |
| | ITE | I | ? | ? | ? | ? | ? |
| PERS | LEGER-DOLS | ? | ? | ? | ? | ? | ? |
| ROLE | AC2 | ? | ? | ? | AUCUN | ? | ON |
| | AFD | ? | ? | ? | AUCUN | ? | ON |
| | ALTAS | ? | ? | ? | AUCUN | ? | 01 |
| | ALTE | ? | ? | ? | AUCUN | ? | 01 |
| | APD | ? | ? | ? | AUCUN | ? | 01 |
| | FFTA | ? | ? | ? | AUCUN | ? | 1N |
| | FLF | ? | ? | ? | AUCUN | ? | ON |
| | IKOCONC | ? | ? | ? | AUCUN | ? | 11 |
| | IKOC1 | ? | ? | ? | AUCUN | ? | ON |
| | IKOC2 | ? | ? | ? | AUCUN | ? | ON |
| | MCON | ? | ? | ? | AUCUN | ? | ON |
| | MLF | ? | ? | ? | AUCUN | ? | ON |
| | MLT | ? | ? | ? | AUCUN | ? | ON |
| | PAF | ? | ? | ? | AUCUN | ? | 11 |
| | PRSP | ? | ? | ? | AUCUN | ? | ON |
| | PRTP | ? | ? | ? | AUCUN | ? | ON |
| | RECON | ? | ? | ? | AUCUN | ? | ON |
| | REUT | ? | ? | ? | AUCUN | ? | ON |
| | SCHIO | ? | ? | ? | AUCUN | ? | ON |
| | SCHRSP | ? | ? | ? | AUCUN | ? | 1N |
| | SCHSTA | ? | ? | ? | AUCUN | ? | ON |
| | SEAF | ? | ? | ? | AUCUN | ? | ON |
| | SETS | ? | ? | ? | AUCUN | ? | ON |
| | TASC1 | ? | ? | ? | AUCUN | ? | ON |
| | TASLTAS | ? | ? | ? | AUCUN | ? | ON |
| | TASRO | ? | ? | ? | AUCUN | ? | 2N |
| | TECONC | ? | ? | ? | AUCUN | ? | ON |
| | TEFTA | ? | ? | ? | AUCUN | ? | ON |
| | TELTE | ? | ? | ? | AUCUN | ? | ON |
| | TERO | ? | ? | ? | AUCUN | ? | ON |
| | TESTA | ? | ? | ? | AUCUN | ? | 11 |
| | TFH | ? | ? | ? | AUCUN | ? | ON |
| | TIO | ? | ? | ? | AUCUN | ? | ON |

08/07/85 ENSEMBLE DES OBJETS INTRODUITS

| TYPE_O | NOM_O | SSTYPE_O | SSCLASSE_O | DATE_O | C6_O | C7_O | C8_O |
|---|---|---|---|---|---|---|---|
| ROLE | TLT | ? | ? | ? | AUCUN | ? | IN |
| | TPH | ? | ? | ? | AUCUN | ? | ON |
| | TRTP | ? | ? | ? | AUCUN | ? | IN |
| | TTS | ? | ? | ? | AUCUN | ? | ON |
| | TUT | ? | ? | ? | AUCUN | ? | ON |
| ROLESP | AFFILI | ? | ? | ? | CREAS | ? | ? |
| | CREASOD | ? | ? | ? | CREAS | ? | ? |
| ROLETP | AS | ? | ? | ? | CREAT | ? | ? |
| | ASE | ? | ? | ? | CREAT | 1 | N |
| | ASEEC | ? | ? | ? | CREAT | 1 | N |
| | ASEEL | ? | ? | ? | CREAT | 1 | N |
| | ASEEM | ? | ? | ? | CREAT | 1 | N |
| | ASEFC | ? | ? | ? | CREAT | 1 | N |
| | ASEFC1 | ? | ? | ? | CREAT | 1 | N |
| | ASEFC2 | ? | ? | ? | CREAT | 1 | N |
| | ASEFL | ? | ? | ? | CREAT | 1 | N |
| | ASEFL1 | ? | ? | ? | CREAT | 1 | N |
| | ASEFL2 | ? | ? | ? | CREAT | 1 | N |
| | ASEFM | ? | ? | ? | CREAT | 1 | N |
| | ASEFM1 | ? | ? | ? | CREAT | 1 | N |
| | ASEFM2 | ? | ? | ? | CREAT | 1 | N |
| | ASEFM3 | ? | ? | ? | CREAT | 1 | N |
| | ASEFM4 | ? | ? | ? | CREAT | 1 | N |
| | ASEFM5 | ? | ? | ? | CREAT | 1 | N |
| | ASEFM6 | ? | ? | ? | CREAT | 1 | N |
| | ASELC | ? | ? | ? | CREAT | 1 | N |
| | ASELL | ? | ? | ? | CREAT | 1 | N |
| | ASELM | ? | ? | ? | CREAT | 1 | N |
| | ASEPL | ? | ? | ? | CREAT | 1 | N |
| | ASEPL1 | ? | ? | ? | CREAT | 1 | N |
| | ASEPL2 | ? | ? | ? | CREAT | 1 | N |
| | ASETC | ? | ? | ? | CREAT | 1 | N |
| | ASETL | ? | ? | ? | CREAT | 1 | N |
| | ASETM | ? | ? | ? | CREAT | 1 | N |
| | ASI | ? | ? | ? | CREAT | 1 | N |
| | ASJ | ? | ? | ? | CREAT | 1 | N |
| | ASM | ? | ? | ? | CREAT | 1 | N |
| | INSATTR1 | ? | ? | ? | CREAT | ? | ? |
| | INSATTR2 | ? | ? | ? | CREAT | ? | ? |
| | INSTAS | ? | ? | ? | CREAT | ? | ? |
| | INSTE | ? | ? | ? | CREAT | ? | ? |
| | INSTRI1 | ? | ? | ? | CREAT | ? | ? |
| | INSTXT | ? | ? | ? | CREAT | ? | ? |
| SCHEMA | AFFILI | SE | ? | ? | T | ? | ? |
| | OD | CONCEPTUEL | ? | 10685 | T | ? | ? |
| SERV | GENERAL | ? | ? | ? | ? | ? | ? |
| TAS | AFFECTE | ? | ? | ? | ? | ? | ? |
| | COMP1 | ? | ? | ? | ? | ? | ? |
| | COMP2 | ? | ? | ? | ? | ? | ? |
| | CONC | ? | ? | ? | ? | ? | ? |
| | CONNECTE | ? | ? | ? | ? | ? | ? |
| | DECOMP | ? | ? | ? | ? | ? | ? |
| | FTA | ? | ? | ? | ? | ? | ? |

08/07/85 ENSEMBLE DES OBJETS INTRODUITS

| TYPE_O | NOM_O | SSTYPE_O | SSCLASSE_O | DATE_O | C6_O | C7_O | C8_O |
|---|---|---|---|---|---|---|---|
| TAS | HIER | ? | ? | ? | ? | ? | ? |
| | I_O | ? | ? | ? | ? | ? | ? |
| | LIEN_TAS | ? | ? | ? | ? | ? | ? |
| | LIEN_TE | ? | ? | ? | ? | ? | ? |
| | LOCF | ? | ? | ? | ? | ? | ? |
| | LOCT | ? | ? | ? | ? | ? | ? |
| | ROLE | ? | ? | ? | ? | ? | ? |
| | ROLESP | ? | ? | ? | ? | ? | ? |
| | ROLETP | ? | ? | ? | ? | ? | ? |
| | STA | ? | ? | ? | ? | ? | ? |
| | TRT_SERV | ? | ? | ? | ? | ? | ? |
| | UTILISE | ? | ? | ? | ? | ? | ? |
| TE | AFFILI | ? | ? | ? | ? | ? | ? |
| | ATTR | ? | ? | ? | ? | ? | ? |
| | FICHIER | ? | ? | ? | ? | ? | ? |
| | IKO | ? | ? | ? | ? | ? | ? |
| | MEM_AUX | ? | ? | ? | ? | ? | ? |
| | PERS | ? | ? | ? | ? | ? | ? |
| | RES | ? | ? | ? | ? | ? | ? |
| | SCHEMA | ? | ? | ? | ? | ? | ? |
| | SERV | ? | ? | ? | ? | ? | ? |
| | TAS | ? | ? | ? | ? | ? | ? |
| | TE | ? | ? | ? | ? | ? | ? |
| | TRT | ? | ? | ? | ? | ? | ? |
| TRT | AS | MODULE | EXECCOBOL | ? | T | 1 | N |
| | ASE | SSMODULE | EXECCOBOL | ? | T | 1 | N |
| | ASEEC | OPERATION | EXECCOBOL | ? | T | 1 | N |
| | ASEEL | OPERATION | EXECCOBOL | ? | T | 1 | N |
| | ASEEM | OPERATION | EXECCOBOL | ? | T | 1 | N |
| | ASEFC | OPERATION | EXECCOBOL | ? | T | 1 | N |
| | ASEFC1 | PGM | EXECCOBOL | ? | T | 1 | N |
| | ASEFC2 | PGM | EXECCOBOL | ? | T | 1 | N |
| | ASEFL | OPERATION | EXECCOBOL | ? | T | 1 | N |
| | ASEFL1 | PGM | EXECCOBOL | ? | T | 1 | N |
| | ASEFL2 | PGM | EXECCOBOL | ? | T | 1 | N |
| | ASEFM | OPERATION | EXECCOBOL | ? | T | 1 | N |
| | ASEFM1 | PGM | EXECCOBOL | ? | T | 1 | N |
| | ASEFM2 | PGM | EXECCOBOL | ? | T | 1 | N |
| | ASEFM3 | PGM | EXECCOBOL | ? | T | 1 | N |
| | ASEFM4 | PGM | EXECCOBOL | ? | T | 1 | N |
| | ASEFM5 | PGM | EXECCOBOL | ? | T | 1 | N |
| | ASEFM6 | PGM | EXECCOBOL | ? | T | 1 | N |
| | ASELC | OPERATION | EXECCOBOL | ? | T | 1 | N |
| | ASELL | OPERATION | EXECCOBOL | ? | T | 1 | N |
| | ASELM | OPERATION | EXECCOBOL | ? | T | 1 | N |
| | ASEPL | OPERATION | EXECCOBOL | ? | T | 1 | N |
| | ASEPL1 | PGM | EXECCOBOL | ? | T | 1 | N |
| | ASEPL2 | PGM | EXECCOBOL | ? | T | 1 | N |
| | ASETC | OPERATION | EXECCOBOL | ? | T | 1 | N |
| | ASETL | OPERATION | EXECCOBOL | ? | T | 1 | N |
| | ASETM | OPERATION | EXECCOBOL | ? | T | 1 | N |
| | ASI | SSMODULE | EXECCOBOL | ? | T | 1 | N |
| | ASJ | SSMODULE | EXECCOBOL | ? | T | 1 | N |
| | ASM | SSMODULE | EXECCOBOL | ? | T | 1 | N |
| | INSATTR1 | PGM | RUNSQL | 10885 | T | 1 | N |
| | INSATTR2 | PGM | RUNSQL | 10885 | T | 1 | N |

08/07/85 ENSEMBLE DES OBJETS INTRODUITS

| TYPE_O | NOM_O | SSTYPE_O | SSCLASSE_O | DATE_O | C6_O | C7_O | C8_O |
|---|---|---|---|---|---|---|---|
| TRT | INSTAS | PGM | RUNSQL | 10885 | T | 1 | N |
| | INSTE | PGM | RUNSQL | 10685 | T | 1 | N |
| | INSTRT1 | PGM | RUNSQL | 10685 | T | 1 | N |

| TYPE1_R | NOM1_R | TYPE2_R | NOM2_R |
|---|---|---|---|
| ATTR | NOM_A | COMP2 | IA |
| | NOM_F | | IF |
| | NOM_IKO | | IIKO |
| | NOM_M | | IM |
| | NOM_P | | IP |
| | NOM_RE | | IRE |
| | NOM_SCH | | ISCH |
| | NOM_SE | | ISE |
| | NOM_T | | IT |
| | NOM_TAS | | ITAS |
| | NOM_TE | | ITE |
| | CONNECT_RO | TAS | ROLE |
| | DMAJ_LF | | LOCF |
| | NOM_C1 | | COMP1 |
| | NOM_C2 | | COMP2 |
| | NOM_IO | | I_O |
| | NOM_LF | | LOCF |
| | NOM_LT | | LOCT |
| | NOM_RO | | ROLE |
| | NOM_RSP | | ROLESP |
| | NOM_RTP | | ROLETP |
| | ROLE_C1 | | COMP1 |
| | ROLE_C2 | | COMP2 |
| | ROLE_IO | | I_O |
| | ROLE_LF | | LOCF |
| | ROLE_LT | | LOCT |
| | ROLE_RO | | ROLE |
| | ROLE_RSP | | ROLESP |
| | ROLE_RTP | | ROLETP |
| | SENS_C1 | | COMP1 |
| | SENS_C2 | | COMP2 |
| | ABF | TE | AFFILI |
| | ABL | | AFFILI |
| | ACR | | AFFILI |
| | ADR | | AFFILI |
| | AFF | | AFFILI |
| | ANR | | AFFILI |
| | ARC | | AFFILI |
| | BTE | | AFFILI |
| | BUR | | AFFILI |
| | CAD | | AFFILI |
| | CAO | | AFFILI |
| | CAT | | AFFILI |
| | CEA | | AFFILI |
| | CFA | | AFFILI |
| | CFC | | AFFILI |
| | CID | | AFFILI |
| | CJT | | AFFILI |
| | CLASSE_A | | ATTR |
| | CLASSE_F | | FICHIER |
| | CLASSE_M | | MEM_AUX |
| | CLASSE_RE | | RES |
| | CMP | | AFFILI |
| | CON | | AFFILI |
| | CRE | | AFFILI |
| | CSA | | AFFILI |

| TYPE1_R | NOM1_R | TYPE2_R | NOM2_R |
|---|---|---|---|
| ATTR | CST | TE | AFFILI |
| | CZR | | AFFILI |
| | DATE_SCH | | SCHEMA |
| | DCR | | AFFILI |
| | DDN_P | | PERS |
| | DEA | | AFFILI |
| | DEV_SCH | | SCHEMA |
| | DEV_T | | TRT |
| | DIG | | AFFILI |
| | DIP | | AFFILI |
| | DMAJ_T | | TRT |
| | DMJ | | AFFILI |
| | DOUBLES_IKO | | IKO |
| | DRA | | AFFILI |
| | DSA | | AFFILI |
| | ECI | | AFFILI |
| | ENR | | AFFILI |
| | ETAPE_T | | TRT |
| | GEN | | AFFILI |
| | INA | | AFFILI |
| | INS | | AFFILI |
| | JNA | | AFFILI |
| | LOC | | AFFILI |
| | MUC | | AFFILI |
| | NAC | | AFFILI |
| | NAT | | AFFILI |
| | NIVEAU_A | | ATTR |
| | NNA | | AFFILI |
| | NOF | | AFFILI |
| | NOL | | AFFILI |
| | NOM_A | | ATTR |
| | NOM_F | | FICHIER |
| | NOM_IKO | | IKO |
| | NOM_M | | MEM_AUX |
| | NOM_P | | PERS |
| | NOM_RE | | RES |
| | NOM_SCH | | SCHEMA |
| | NOM_SE | | SERV |
| | NOM_T | | TRT |
| | NOM_TAS | | TAS |
| | NOM_TE | | TE |
| | NUM | | AFFILI |
| | ORDRE_IKO | | IKO |
| | ORR | | AFFILI |
| | PAN | | AFFILI |
| | PHO | | AFFILI |
| | PLC | | AFFILI |
| | PPE | | AFFILI |
| | PRF | | AFFILI |
| | PRIV_SCH | | SCHEMA |
| | PRIV_T | | TRT |
| | PRO | | AFFILI |
| | QUA | | AFFILI |
| | RAF | | AFFILI |
| | REV | | AFFILI |
| | RLI | | AFFILI |
| | RRP | | AFFILI |
| | SEA | | AFFILI |

| TYPE1_R | NOM1_R | TYPE2_R | NOM2_R |
|---|---|---|---|
| ATTR | SEX | TE | AFFILI |
| | SRP | | AFFILI |
| | SSA | | AFFILI |
| | TEA | | AFFILI |
| | TSA | | AFFILI |
| | TYPE_A | | ATTR |
| | TYPE_F | | FICHIER |
| | TYPE_IKO | | IKO |
| | TYPE_M | | MEM_AUX |
| | TYPE_RE | | RES |
| | TYPE_SCH | | SCHEMA |
| | TYPE_T | | TRT |
| | VERS_SCH | | SCHEMA |
| | VERS_T | | TRT |
| COMP2 | IA | IKO | IA |
| | IF | | IF |
| | IIKO | | IIKO |
| | IM | | IM |
| | IP | | IP |
| | IRE | | IRE |
| | ISCH | | ISCH |
| | ISE | | ISE |
| | IT | | IT |
| | ITAS | | ITAS |
| | ITE | | ITE |
| FICHIER | AFFILI | TE | AFFILI |
| I_O | INSATTR1DD | SCHEMA | DD |
| | INSATTR2DD | | DD |
| | INSTASDD | | DD |
| | INSTEDD | | DD |
| | INSTRT1DD | | DD |
| | INSTXTDD | | DD |
| | INSATTR1DD | TRT | INSATTR1 |
| | INSATTR2DD | | INSATTR2 |
| | INSTASDD | | INSTAS |
| | INSTEDD | | INSTE |
| | INSTRT1DD | | INSTRT1 |
| | INSTXTDD | | INSTXT |
| IKO | IA | TE | ATTR |
| | IF | | FICHIER |
| | IIKO | | IKO |
| | IM | | MEM_AUX |
| | IP | | PERS |
| | IRE | | RES |
| | ISCH | | SCHEMA |
| | ISE | | SERV |
| | IT | | TRT |
| | ITAS | | TAS |
| | ITE | | TE |
| PERS | LEGER-DOLS | ROLESP | CREASDD |
| | PCAS | | AFFILI |

| TYPE1_R | NOM1_R | TYPE2_R | NOM2_R |
|---|---|---|---|
| PERS | LEGER-DOLS | ROLETP | INSATTR1 |
| | LEGER-DOLS | | INSATTR2 |
| | LEGER-DOLS | | INSTAS |
| | LEGER-DOLS | | INSTE |
| | LEGER-DOLS | | INSTRT1 |
| | LEGER-DOLS | | INSTXT |
| | PCAS | | AS |
| | LEGER-DOLS | SERV | GENERAL |
| | PCAS | TRT | ASE |
| | PCAS | | ASEEC |
| | PCAS | | ASEEL |
| | PCAS | | ASEEM |
| | PCAS | | ASEFC |
| | PCAS | | ASEFC1 |
| | PCAS | | ASEFC2 |
| | PCAS | | ASEFL |
| | PCAS | | ASEFL1 |
| | PCAS | | ASEFL2 |
| | PCAS | | ASEFM |
| | PCAS | | ASEFM1 |
| | PCAS | | ASEFM2 |
| | PCAS | | ASEFM3 |
| | PCAS | | ASEFM4 |
| | PCAS | | ASEFM5 |
| | PCAS | | ASEFM6 |
| | PCAS | | ASELC |
| | PCAS | | ASELL |
| | PCAS | | ASELM |
| | PCAS | | ASEPL |
| | PCAS | | ASEPL1 |
| | PCAS | | ASEPL2 |
| | PCAS | | ASETC |
| | PCAS | | ASETL |
| | PCAS | | ASETM |
| | PCAS | | ASI |
| | PCAS | | ASJ |
| | PCAS | | ASM |

| ROLE | AC2 | TAS | COMP2 |
|---|---|---|---|
| | AFD | | DECOMP |
| | ALTAS | | LIEN_TAS |
| | ALTE | | LIEN_TE |
| | APD | | DECOMP |
| | FFTA | | FTA |
| | FLF | | LOCF |
| | IKOCONC | | CONC |
| | IKOC1 | | COMP1 |
| | IKOC2 | | COMP2 |
| | MCON | | CONNECTE |
| | MLF | | LOCF |
| | MLT | | LOCT |
| | PAF | | AFFECTE |
| | PRSP | | ROLESP |
| | PRTP | | ROLETP |
| | RECON | | CONNECTE |
| | REUT | | UTILISE |

| TYPE1_R | NOM1_R | TYPE2_R | NOM2_R |
|---|---|---|---|
| ROLE | SCHIO | TAS | I_O |
| | SCHRSP | | ROLESP |
| | SCHSTA | | STA |
| | SEAF | | AFFECTE |
| | SETS | | TRT_SERV |
| | TASC1 | | COMP1 |
| | TASLTAS | | LIEN_TAS |
| | TASRO | | ROLE |
| | TECONC | | CONC |
| | TEFTA | | FTA |
| | TELTE | | LIEN_TE |
| | TERO | | ROLE |
| | TESTA | | STA |
| | TFH | | HIER |
| | TIO | | I_O |
| | TLT | | LOCT |
| | TPH | | HIER |
| | TRTP | | ROLETP |
| | TTS | | TRT_SERV |
| | TUT | | UTILISE |
| | AC2 | TE | ATTR |
| | AFD | | ATTR |
| | ALTAS | | ATTR |
| | ALTE | | ATTR |
| | APD | | ATTR |
| | FFTA | | FICHIER |
| | FLF | | FICHIER |
| | IKOCONC | | IKO |
| | IKOC1 | | IKO |
| | IKOC2 | | IKO |
| | MCON | | MEM_AUX |
| | MLF | | MEM_AUX |
| | MLT | | MEM_AUX |
| | PAF | | PERS |
| | PRSP | | PERS |
| | PRTP | | PERS |
| | RECON | | RES |
| | REUT | | RES |
| | SCHIO | | SCHEMA |
| | SCHRSP | | SCHEMA |
| | SCHSTA | | SCHEMA |
| | SEAF | | SERV |
| | SETS | | SERV |
| | TASC1 | | TAS |
| | TASLTAS | | TAS |
| | TASRO | | TAS |
| | TECONC | | TE |
| | TEFTA | | TE |
| | TELTE | | TE |
| | TERO | | TE |
| | TESTA | | TE |
| | TFH | | TRT |
| | TIO | | TRT |
| | TLT | | TRT |
| | TPH | | TRT |
| | TRTP | | TRT |
| | TTS | | TRT |

| TYPE1_R | NOM1_R | TYPE2_R | NOM2_R |
|---|---|---|---|
| ROLE | TUT | TE | TRT |
| ROLESP | AFFILI | SCHEMA | AFFILI |
| | CREASDD | | DD |
| ROLETP | AS | TRT | AS |
| | ASE | | ASE |
| | ASEEC | | ASEEC |
| | ASEEL | | ASEEL |
| | ASEEM | | ASEEM |
| | ASEFC | | ASEFC |
| | ASEFC1 | | ASEFC1 |
| | ASEFC2 | | ASEFC2 |
| | ASEFL | | ASEFL |
| | ASEFL1 | | ASEFL1 |
| | ASEFL2 | | ASEFL2 |
| | ASEFM | | ASEFM |
| | ASEFM1 | | ASEFM1 |
| | ASEFM2 | | ASEFM2 |
| | ASEFM3 | | ASEFM3 |
| | ASEFM4 | | ASEFM4 |
| | ASEFM5 | | ASEFM5 |
| | ASEFM6 | | ASEFM6 |
| | ASELC | | ASELC |
| | ASELL | | ASELL |
| | ASELM | | ASELM |
| | ASEPL | | ASEPL |
| | ASEPL1 | | ASEPL1 |
| | ASEPL2 | | ASEPL2 |
| | ASETC | | ASETC |
| | ASETL | | ASETL |
| | ASETM | | ASETM |
| | ASI | | ASI |
| | ASJ | | ASJ |
| | ASM | | ASM |
| | INSATTR1 | | INSATTR1 |
| | INSATTR2 | | INSATTR2 |
| | INSTAS | | INSTAS |
| | INSTE | | INSTE |
| | INSTRT1 | | INSTRT1 |
| | INSTXT | | INSTXT |
| SCHEMA | AFFILI | TE | AFFILI |
| | DD | | ATTR |
| | DD | | FICHIER |
| | DD | | IKO |
| | DD | | MEM_AUX |
| | DD | | PERS |
| | DD | | RES |
| | DD | | SCHEMA |
| | DD | | SERV |
| | DD | | TAS |
| | DD | | TE |
| | DD | | TRT |
| SERV | AAAAAAAAAAA | TRT | INSTE |
| | CAS | | AS |
| | GENERAL | | INSATTR1 |

08/07/85 ENSEMBLES DES RELATIONS INTRODUITES

| TYPE1_R | NOM1_R | TYPE2_R | NOM2_R |
|---|---|---|---|
| SERV | GENERAL | TRT | INSATTR2 |
| | GENERAL | | INSTAS |
| | GENERAL | | INSTE |
| | GENERAL | | INSTRT1 |
| | GENERAL | | INSTXT |
| TRT | ASE | TRT | AS |
| | ASEEC | | ASE |
| | ASEEL | | ASE |
| | ASEEM | | ASE |
| | ASEFC | | ASE |
| | ASEFC1 | | ASEFC |
| | ASEFC2 | | ASEFC |
| | ASEFL | | ASE |
| | ASEFL1 | | ASEFL |
| | ASEFL2 | | ASEFL |
| | ASEFM | | ASE |
| | ASEFM1 | | ASEFM |
| | ASEFM2 | | ASEFM |
| | ASEFM3 | | ASEFM |
| | ASEFM4 | | ASEFM |
| | ASEFM5 | | ASEFM |
| | ASEFM6 | | ASEFM |
| | ASELC | | ASE |
| | ASELL | | ASE |
| | ASELM | | ASE |
| | ASEPL | | ASE |
| | ASEPL1 | | ASEPL |
| | ASEPL2 | | ASEPL |
| | ASETC | | ASE |
| | ASETL | | ASE |
| | ASETM | | ASE |
| | ASI | | AS |
| | ASJ | | AS |
| | ASM | | AS |

# BIBLIOGRAPHIE.

## ARTICLES ET PUBLICATIONS


1) DDSWP (Data Dictionary Systems Working Party)
British Computer Society
JOURNAL OF DEVELOPMENT Summer 82.

2) A Survey of Data Dictionaries
DATAMATION March 81.

3) BODART F.
Proposal for a standard Description of Computer Applications
OECD Technical Cooperation Service.

4) MAYNE Alan
The central Role of Dictionary Systems in Information Resource Management
Perganon Infotech. December 82.

5) The integrated Dictionary/Directory System
Computing Surveys Vol. 14 No. 2 June 82.

6) MAYNE Alan
The Architecture of a Dictionary System
ISO February 83.

7) VAN LAMSWEERDE A.
Les Outils d'Aide au Développement de Logiciels: Un aperçu des tendances actuelles
XVèmes journées internationales de l'informatique et de l'automatisme.
Paris 16-18 juin 82.

8) TEICHROEW, RATAS, HERSHEY
An Introduction to Computer-aided Documentation of User Requirements for Computer-based Information processing Systems.
Informations Systems and Organisation Structure.
Berlin 75.

THESES ET MEMOIRES

9) PIGNEUR Y.
Synthèse de Propositions de Spécifications théoriques d'un Systeme de Documentation.
FUNDP Institut d'Informatique 77.

10) ZIMMER H.
Contenu et Usages des Dictionnaires de Données.
FUNDP Institut d'Informatique 82.

OUVRAGES ET COURS

11) BODART F., PIGNEUR Y.
Conception assistée des Applications informatiques.
Presses Universitaires de Namur.
Masson 83.

12) CLARINVAL A.
Comprendre, Connaitre et Maitriser le COBOL.
Presses Universitaires de Namur.82.

13) HAINAUT J.L.
Cours de Conception de Bases de Données.
FUNDP Namur Institut d'Informatique.

14) HAINAUT J.L.
Cours de Technologie de Fichiers.
FUNDP Namur Institut d'Informatique.

15) HAINAUT J.L.
Cadre de Référence.
FUNDP Namur Institut d'Informatique.

16) HAINAUT J.L.
Le Modèle relationnel.
FUNDP Namur Institut d'Informatique.

17) HAINAUT J.L.
Le Modèle d'Accès Généralisé.
FUNDP Namur Institut d'Informatique.